ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Realisierung und Evaluierung von Interaktionskonzepten für elektronische Fragebögen auf Smartwatches

Masterarbeit an der Universität Ulm

Vorgelegt von:
Martin Heinzelmann
martin.heinzelmann@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

Betreuer:
Marc Schickler

2016

Fassung 1. April 2016

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Der Begriff Tinnitus bezeichnet Töne oder Geräusche, die eine Person wahrnimmt, jedoch keiner äußeren physikalischen Quelle zugeordnet werden können. Der subjektive Tinnitus wird nur vom betroffenen Menschen wahrgenommen und kann mit Messgeräten nicht erfasst und aufgezeichnet werden.

Das *TrackYourTinnitus* Projekt dient zur Erfassung und Aufzeichnung dieser krankheitsbedingten, subjektiven Lautstärkeschwankungen. Zur Erfassung der Daten werden mobile Anwendungen für *Smartphones* eingesetzt, die den Projektteilnehmern zu bestimmten Zeitpunkten Fragebögen bereitstellen. Diese sollen von den Teilnehmern zeitnahe bearbeitet werden. Die erfassten Daten werden auf einem Server aufbereitet und gespeichert, sodass sie später unterstützend für die Behandlung von Patienten und nutzbringend für die Forschung eingesetzt werden können.

Um die Akzeptanz der entwickelten mobilen Anwendungen zu steigern, soll in dieser Masterarbeit geprüft werden, ob sich *Smartwatches* für den Einsatz in einem *Mobile Crowd Sensing System* eignen. Hierfür wurde ein Umfragesystem mit *Smartwatch* Unterstützung implementiert. In die *Smartwatch*-Anwendung wurden verschiedene Eingabeformen zur Beantwortung von Fragen integriert. Die entwickelten Eingabeformen zielen auf eine komfortable Beantwortung von Umfragen über den Touchscreen und die integrierte Spracheingabe. Diese werden später in einer Studie auf ihre Tauglichkeit evaluiert. Für die Durchführung eines Vergleiches zwischen *Smartwatch* und *Smartphone* wurde zudem eine *Smartphone*- Anwendung implementiert. Ziel der Studie ist es, festzustellen ob der Einsatz von *Smartwatches* in einem *Mobile Crowd Sensing System* sinnvoll ist.

Danksagung

An dieser Stelle möchte ich mich bei allen Unterstützern und Unterstützerinnen bedanken, die zum Gelingen dieser Masterarbeit beigetragen haben. Mein Dank gilt Prof. Dr. Manfred Reichert für die Möglichkeit, zu diesem interessanten Themengebiet einen Beitrag leisten zu können.

Ein besonderer Dank gilt meinem Betreuer Marc Schickler für die Unterstützung bei der Themenfindung und der anschließenden Betreuung der Masterarbeit.

Gerne möchte ich mich bei Dr. Rüdiger Pryss und Dr. Winfried Schlee für die persönliche und fachliche Unterstützung bedanken.

Es war mir eine Freude in so einem professionellen Umfeld studieren und arbeiten zu dürfen.

Vielen Dank auch an alle Studienteilnehmer und Studienteilnehmerinnen, die gleichfalls zum Erfolg dieser Abschlussarbeit beigetragen haben.

Darüber hinaus möchte ich mich bei meiner Familie bedanken, die mich während meines gesamten Studiums in jeglicher Form unterstützt hat.

Abschließend möchte ich mich auch bei all meinen Freunden bedanken, die mir während der Studienzeit immer zur Seite standen.

Inhaltsverzeichnis

1

Einführung

Mit der Einführung des *iPhones* im September 2007 entfachte *Apple* eine grundlegende Veränderung unserer mobilen Medienlandschaft. *Apple* stellte mit dem *iPhone* nicht nur ein Telefon vor, sondern einen ganzen Taschencomputer, mit dem man zudem noch SMS und E-Mails schreiben, sowie im Internet surfen konnte. Seither sind auch andere Unternehmen wie beispielsweise *Google* und *Microsoft* in diesen Markt eingestiegen und bieten ein vielseitiges Portfolio an *Smartphones* und *Tablets* an. Über die Jahre wurden die mobilen Geräte immer leistungsstärker und ausgereifter, was Anwendungen bis hin in den *Augmented Reality* Bereich (siehe *AREA*[1]) möglich machte. Im Jahr 2014 nutzten weltweit 1,59 Milliarden Menschen ein *Smartphone*. In diesem Jahr werden rund 2,08 Milliarden *Smartphone* Nutzer erwartet [37]. *Smartphones* und *Tablets* sind somit aus unserem privaten und beruflichen Leben nicht mehr wegzudenken.

Seit einiger Zeit, ist eine weitere Generation von mobilen Geräten dabei, sich am Markt zu etablieren. Die sogenannten *Wearables*. Der Begriff *Wearable* ist als Oberbegriff zu verstehen und bezeichnet mobile Computersysteme, die während der Nutzung am Körper des Benutzers angebracht sind. Diese *Wearables* unterscheiden sich von *Smartphones* und *Tablets* dadurch, dass sie wesentlich kleiner sind und nicht im Fokus des Benutzers stehen. Eine Untergruppe mit der sich diese Arbeit genauer beschäftigt sind die sogenannten *Smartwatches*. Als *Smartwatches* werden Armbanduhren bezeichnet, die über zusätzliche Computerfunktionalität, Kommunikationstechnik und Sensoren verfügen. Diese können am Handgelenk getragen und mit dem *Smartphone* verbunden werden.

[1] *AREA* steht für *Augmented Reality Engine Application*. Dabei handelt es sich um eine Anwendung, mit der sich der Nutzer Sehenswürdigkeiten in der Umgebung anzeigen lassen kann [5, 32].

Der Umstand, dass heutzutage beinahe jeder Mensch ein *Smartphone* oder vielleicht auch schon eine *Smartwatch* mit sich trägt, bietet auch der Medizin ein breites Spektrum zur begleitenden Therapie von Erkrankungen. Mit der Erfassung verschiedener relevanter Daten mit Hilfe dieser mobilen Geräte, können Krankheitsverläufe einfach aufgezeichnet und nachvollzogen werden. Schon im Jahr 2014 hat das Institut für Datenbanken und Informationssysteme der Universität Ulm in Zusammenarbeit mit der *Tinnitus Research Initiative* (TRI) ein System entwickelt, dass es tinnitusgeschädigten Patienten erlaubt, ihren Krankheitsverlauf zu dokumentieren. Dieses System mit dem Namen *TrackYourTinnitus* stützt sich auf ein *Mobile Crowd Sensing System*, dass es den Patienten möglich macht, ihren Krankheitsverlauf mit ihrem eigenen *Smartphone* oder *Tablet* aufzuzeichnen und zu dokumentieren [23].

Im nachfolgenden Abschnitt wird aufgezeigt, was sich hinter der Krankheit Tinnitus verbirgt. Darauf folgend wird das *TrackYourTinnitus* Projekt in seiner Funktionsweise beschrieben und aufgezeigt wie die neue Generation von *Wearables* im *TrackYourTinnitus* Projekt eingesetzt werden können. Abschließend wird die Zielsetzung dieser Arbeit vorgestellt und der Aufbau der Arbeit erläutert.

1.1 Tinnitus

Der Begriff Tinnitus beschreibt ein Krankheitsbild, bei dem der Patient dauerhaft Töne oder Ohrgeräusche wahrnimmt, die keiner externen Schallquelle zugeordnet werden können. Diese Töne oder Ohrgeräusche werden von den Patienten meist als Pfeifen oder Rauschen beschrieben. Der Tinnitus kann in objektiven und subjektiven Tinnitus kategorisiert werden. Der objektive Tinnitus beschreibt solche Ohrgeräusche, die an anderen Körperstellen erzeugt und über das Ohr wahrgenommen werden. Diese Form des Tinnitus kommt sehr selten vor und kann gut behandelt werden [33].

Der subjektive Tinnitus tritt hingegen häufiger auf. Er kann in verschiedenen Formen, von kaum wahrnehmbar bis hin zur chronischen Dauerbelastung, auftreten. Die Wahrnehmung des subjektiven Tinnitus unterliegt gewiesen Schwankungen. Das bedeutet, dass ein Patient den Tinnitus von einem Moment zum Nächsten unterschiedlich stark

wahrnehmen kann. Von einem chronischen Tinnitus spricht man bei Patienten die länger als 6 Monate an einem Tinnitus leiden. Auch hier können Schwankungen bei der Tinnituswahrnehmung auftreten [33].

In vielen Fällen tritt der subjektive Tinnitus in Verbindung mit dem Verlust des Hörvermögens auf [28]. Beim subjektiven Tinnitus existiert keine erkennbare Schallquelle und es gibt derzeit kein objektives Maß für die Messung des Tinnitus. Es ist nur dem Patienten möglich, den Tinnitus in seiner Form und Wahrnehmung zu beschreiben [29].

1.2 TrackYourTinnitus

Die *TrackYourTinnitus* Plattform wurde von der *Tinnitus Research Initiative* (TRI) und dem Institut für Datenbanken und Informationssysteme (DBIS) der Universität Ulm entwickelt [2]. Es wurde entwickelt, um den Tinnitus im Alltag unter Berücksichtigung von täglichen Schwankungen zu dokumentieren. Der Tinnitus ist medizinisch wenig erforscht und neue Erkenntnisse werden eindringlich gesucht. Zu diesem Zweck sind jedoch Daten von Patienten, aus deren Alltag, erforderlich. Bisher war die Erfassung solcher Daten sehr schwierig [33].

Um Daten über das Krankheitsbild Tinnitus zu sammeln, wurden bislang Tinnituspatienten mittels papierbasierten Fragebögen befragt. Der Einsatz von papierbasierten Fragebögen erfordert einen massiven Aufwand bei der Verarbeitung und Analyse der gesammelten Daten. Zudem kann es bei der Übertragung auf elektronische Arbeitsblätter zu Übertragungsfehlern kommen [36, 35]. Der Einsatz von mobilen Geräten wie beispielsweise *Smartphones* bietet an dieser Stelle neue Möglichkeiten. Es können Daten in großer Menge und in kurzer Zeit erfasst werden [33].

Die *TrackYourTinnitus* Plattform setzt *Smartphones* in Verbindung mit einem *Mobile Crowd Sensing System* ein um Schwankungen des Tinnitus aufzuzeichnen. Dabei können Tinnituspatienten die Schwankungen ihres Tinnitus systematisch, über das Ausfüllen spezieller Fragebögen, erfassen [33, 2]. Die ermittelten Daten werden auf einem Server gespeichert und verarbeitet. Die Patienten können später über das Internetportal von

TrackYourTinnitus die erfassten Daten in aufbereiteter Form einsehen und beispielsweise mit ihren Ärzten oder Therapeuten besprechen [2].

1.3 Motivation

Die Qualität der Auswertung und die Rückschlüsse, die daraus gezogen werden, hängen entscheidend von der erfassten Datenmenge ab. Deshalb bekommen die Benutzer von *TrackYourTinnitus* täglich acht Fragebögen auf ihr *Smartphone* zugestellt, die sie zeitnah bearbeiten sollen. Dieser Aufwand kann für den Benutzer lästig erschienen, was eine reduzierte Anzahl an beantworteten Fragebögen zur Folge haben kann. Um diesem Problem entgegen zu wirken, erscheint der Einsatz von *Smartwatches* sinnvoll. Die Bearbeitung der Fragebögen auf der *Smartwatch* soll die Dateneingabe vereinfachen und zu einer größeren Akzeptanz der *TrackYourTinnitus* Plattform führen, so die Hypothese.

1.4 Zielsetzung

Die Aufgabe dieser Masterarbeit besteht darin, die Eignung von *Smartwatches* für den Einsatz in einem *Mobile Crowd Sensing System* zu überprüfen. Dazu gehört die Konzeption und Realisierung einer Anwendung für eine *Smartwatch*. Die zu entwickelnde Anwendung setzt die Realisierung eines *Mobile Crowd Sensing Systems* mit dem Schwerpunkt der Beantwortung von Fragebögen voraus. Außerdem sollen verschiedene Eingabeformen für die Beantwortung von einzelnen Fragen konzipiert und später evaluiert werden. Zum Zeitpunkt dieser Arbeit konnten noch keine veröffentlichten Untersuchungen über den Einsatz von *Smartwatches* in *Mobile Crowd Sensing Systemen* ermittelt werden. Daher beabsichtigt diese Arbeit die Durchführung einer Studie, mit dem Ziel die Eignung von *Smartwatches* für *Mobile Crowd Sensing Systeme* festzustellen und diese anschließend zu publizieren.

1.5 Aufbau der Arbeit

Die vorliegende Masterarbeit setzt sich aus sieben Kapiteln zusammen.

In Kapitel 2 werden die zur Realisierung erforderlichen Grundlagen zusammengefasst. Dies beinhaltet ebenso die Behandlung von wesentlichen Begrifflichkeiten wie beispielsweise die von *Mobile Crowd Sensing*. Zudem erläutert es die Architektur der *TrackYourTinnitus* Plattform. Im Weiteren wird auf die verwendeten Betriebssysteme *Android* und *Android Wear* eingegangen.

Das Kapitel 3 beschreibt die Anforderungsanalyse für das zu entwickelnde System. Dabei werden Anwendungsfälle für die zu entwickelnden Anwendungen vorgestellt und die daraus resultierenden Anforderungen in funktionale und nicht funktionale Anforderungen unterteilt.

Im Anschluss daran folgt in Kapitel 4 die Vorstellung der Architektur des Gesamtsystems. Diese dient als Vorlage für die in Kapitel 5 beschriebene Implementierung. An dieser Stelle werden die Konzepte und die technische Umsetzung der implementierten Anwendungen beschrieben.

Für die Beurteilung des entwickelten Systems wurde eine Studie durchgeführt, die in Kapitel 6 beschrieben wird. Dort werden neben den Studienergebnissen auch der Aufbau und die Durchführung beschrieben.

Das letzte Kapitel 7 fasst die erzielten Ergebnisse diese Masterarbeit nochmals zusammen und gibt einen Ausblick auf mögliche Erweiterungen in der Zukunft.

2

Grundlagen

In diesem Kapitel werden die Grundlagen beschreiben, welche für das Verständnis dieser Arbeit erforderlich sind. Zudem wird das Konzept von *Mobile Crowd Sensing Systemen* und der Einsatz eines solchen im *TrackYourTinnitus* Projekt erläutert. Für diese Arbeit wichtige Protokolle benannt und auf die mobilen Betriebssysteme *Android* und *Android Wear* eingegangen.

2.1 Mobile Crowd Sensing

Unter dem Begriff *Mobile Crowd Sensing* versteht man das Sammeln von großen Datenmengen, unter Einbeziehung einer großen Menschenmenge, mittels Einsatz mobiler Geräte. Durch die Verwendung mobiler Geräte wie *Smartphones*, ist es einem *Mobile Crowd Sensing System* leicht möglich, eine große Anzahl an Menschen zu erreichen, die auf anderen Wegen nur schwer zu ermitteln sind. Um die gesammelten Daten auswerten zu können, werden die Daten auf einem zentralen Server gespeichert und verwaltet. Nachfolgend wird der Aufbau des *Mobile Crowd Sensing Systems* von *TrackYourTinnitus* beschrieben [31].

2.2 TrackYourTinnitus Architektur

Für die Erfassung von Daten unterstützt die *TrackYourTinnitus* Plattform zum Zeitpunkt dieser Arbeit die mobilen Betriebssysteme *Android* und *Apple iOS*. Über die Webseite[1]

[1] `http://www.trackyourtinnitus.org/`

kann auf die *TrackYourTinnitus* Plattform zugegriffen werden. Zur Verarbeitung und Speicherung der gesammelten Daten benutzt die Plattform einen Webservice als sogenanntes *Backend*[2]. Hier werden die Daten in einer MySQL-Datenbank gespeichert. Die Abbildung 2.1 zeigt die Architektur von *TrackYourTinnitus* graphisch auf. Die mobilen Anwendungen für *Apple iOS* und *Android* greifen über die *REST*-Schnittstelle[3] auf das *Backend* zu. Der Datenaustausch zwischen der Webseite und dem *Backend* findet über *Sockets* statt [2, 30].

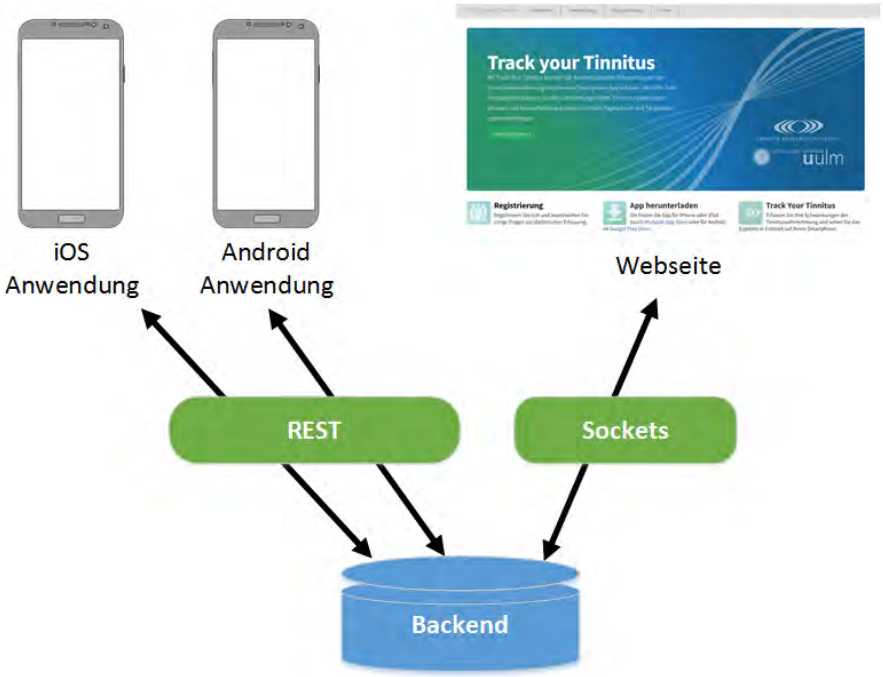

Abbildung 2.1: *TrackYourTinnitus* Architektur [2]

2.3 HTTP

Das *Hypertext Transfer Protocol* (*HTTP*) ist ein generisches und zustandsloses Kommunikationsprotokoll für verteilte Systeme. Es arbeitet auf der Anwendungsschicht des

[2] Ein Backend bezeichnet den Teil einer Software-Anwendung, der sich im Hintergrund mit der Datenverarbeitung beschäftigt [6].
[3] *REST* steht für *Representational State Transfer* [38]

ISO-OSI-Schichtenmodells und wird seit 1990 im *World Wide Web* zum Datentransfer eingesetzt. Ein Datentransfer über *HTTP* wird immer von einem Client initiiert und besteht aus einer Anfrage und einer darauf folgenden Antwort. Hierfür wird die vom *HTTP* Standard definierte Methode `GET` verwendet [39]. Oft wird *HTTP* auch eingesetzt, um sogenannte *RESTful Web Services* zu realisieren. *REST* beschreibt einen Architekturstil für den Aufbau von Webanwendungen. *REST* definiert dabei Prinzipien die bei der Entwicklung einer Webanwendung eingehalten werden müssen. Unter *REST* wird alles als *Ressource* angesehen. Mittels der im *HTTP* Standard definierten Methoden `GET`, `PUT`, `POST` und `DELETE` können Ressourcen abgefragt, gespeichert, geändert oder gelöscht werden [38].

REST besticht mit seiner losen Kopplung, Interoperabilität, guter Wiederverwendbarkeit, Performance und Skalierbarkeit. Durch die Verwendung von *HTTP* als Protokoll und *URIs*[4] als Identifikationsmechanismus wird beinahe jedes Umfeld unterstützt. *REST* bietet damit eine Möglichkeit zur Kommunikation von technisch stark unterschiedlich implementierten Systemen. Durch die Implementierung der von *HTTP* standardisierten Methoden, stellt eine Anwendung eine einheitliche Schnittstelle bereit. Jeder Client, der diese Schnittstelle verwenden kann, ist in der Lage mit weltweit jeder Webanwendung auf Basis von *REST* zu kommunizieren. Bei der Skalierbarkeit von Webanwendung können *REST* und *HTTP* ihre stärken voll ausspielen. Der Umstand, das *HTTP* keinen Zustand speichert macht es möglich, dass aufeinander folgende Anfragen eines Clients nicht vom selben Server beantwortet werden müssen [38].

2.4 JSON

Die *JavaScript Object Notation*, kurz *JSON* ist ein leichtgewichtiges Format zum Datenaustausch auf Anwendungsebene. Es überzeugt durch eine besonders gute Lesbarkeit für den Menschen und einer einfachen Möglichkeit zur Codierung und Decodierung durch eine Maschine. Es handelt sich um ein Format, welches die gängigen Konventionen der C-basierten Programmiersprachen berücksichtigt. Konkret differenziert *JSON*

[4] *URI* steht für *Uniform Resource Identifier* [39]

in die zwei elementaren Grundtypen, Objekte und geordnete Listen. Analog zu den aus den Programmiersprachen bekannten Objekten existieren in *JSON* auch Objekte, welche eine beliebige Anzahl von Name-Wert-Paare beinhalten dürfen. Zur Abbildung von Arrays, Vektoren, Listen oder Sequenzen bietet die Sprache eine geordnete Liste von Werten [24].

2.5 Bluetooth

Als *Bluetooth* wird eine international standardisierte, drahtlose Kommunikationsverbindung bezeichnet. Mit ihr ist es möglich Daten zwischen zwei Geräten auszutauschen. *Bluetooth* ist für Übertragung über kurze Strecken von bis zu 10 Metern geeignet. Mit speziellen Adaptern können aber auch bis zu 100 Meter überbrückt werden. Eingesetzt wird *Bluetooth* vor allem zur drahtlosen Verbindung von Freisprecheinrichtungen, Druckern oder Lautsprechern mit mobilen Endgeräten wie *Smartphones*, *Tablets* oder Notebooks. *Bluetooth* arbeitet im 2,4 GHz *ISM-Band*[5] [22, 26].

2.5.1 Bluetooth Low Energy

Im Jahr 2009 wurde der *Bluetooth* Standard in der Version 4.0 mit *Bluetooth Low Energy* oder kurz *Bluetooth LE* erweitert. Ziel dieser Erweiterung ist es, den Stromverbrauch für eine *Bluetooth*- Verbindung deutlich zu reduzieren. *Bluetooth LE* arbeitet mit einem komplett neu entwickelten *Stack*. Das bedeutet, dass ältere Geräte mit herkömmlichem *Bluetooth* nicht mehr von *Bluetooth LE* unterstützt werden. Module mit *Bluetooth 4.0* benötigen nur wenig Energie und können selbst mit kleinen Batterien mehrere Jahre arbeiten. Dies wird durch die Verlagerung von bestimmten Funktionen vom *Host* in den *Controller* herbei geführt. Durch den geringen Energieverbrauch ist diese Technik hervorragend als Kommunikationsverbindung für *Wearables* geeignet [26].

[5] *ISM-Band* steht für *Industrial, Scientific and Medical Band* [22]

2.6 Android

Das Betriebssystem *Android* ist eine von *Google* am 12. November 2007 vorgestellte Plattform für mobile Endgeräte wie beispielsweise Mobiltelefone und *Tablets*. Im Gegensatz zu dem, kurze Zeit zuvor vorgestellten Betriebssystem *iOS* von *Apple*, veröffentlichte *Google* seine Plattform als *Open Source*. Dadurch war es anderen Unternehmen und Entwicklern möglich, die *Android* Plattform für ihre Zwecke anzupassen und zu erweitern [25, 7].

Für Entwickler stellte *Google* das *Android Software Development Kit* oder kurz *Android SDK* bereit. Interessierte Entwickler können damit bis heute Anwendungen für *Android* entwickeln. Ein weiterer Meilenstein war die Einführung des hauseigenen *Android Market* bzw. des heutigen *Play Store* im Oktober 2008. Fortan konnten Entwickler Anwendungen den Kunden direkt auf ihren Geräten zum Kauf und Herunterladen anbieten. Die individuelle Erweiterung der Funktionen mobiler Endgeräte wurde möglich [25, 7].

Google hat sein Betriebssystem immer weiter entwickelt. In der heutigen Version 6 kann das *Google* Betriebssystem außer in *Smartphones* und *Tablets* auch in anderen Bereichen eingesetzt werden. So gibt es die Ableger *Android Wear*, *Android TV* und *Android Auto*. Dabei ist *Android Wear* für den Einsatz in *Smartwatches* und *Android TV* für Fernseher gedacht. Mit *Android Auto* stellt *Google* ein System für den Einsatz in Entertainment-Systemen für Fahrzeuge zur Verfügung [25, 7].

In den nachfolgenden Abschnitten 2.6.1 bis 2.6.6 wird auf die Grundkomponenten von *Android* eingegangen.

2.6.1 Activity

Die *Activity* stellt die zentrale Komponente in *Android* dar. Aus Sicht des Benutzers verkörpert eine *Activity* eine Funktion einer Anwendung. Aus Sicht des Entwicklers bildet sie jedoch eine klassische Grundkomponente mit einer Benutzeroberfläche. Die Benutzeroberfläche besteht aus einem Fenster, das im Normalfall den ganzen Bildschirm ausfüllt. Innerhalb dieses Fensters können ihre Bedienelemente beliebig angeordnet

werden. Eine App besteht meistens aus mehr als einer *Activity*. In welcher Reihenfolge diese nacheinander aufgerufen werden, kann von den Aktionen des Benutzers abhängen. Wenn beispielsweise in einer Kalenderanwendung alle Termine in einer Liste angezeigt werden, führt das Antippen eines Termins unmittelbar zur Detailansicht dieses Termins. Hiebei wechselt der Benutzer nicht bewusst zwischen einzelnen Aktivities, sondern die Anwendung übernimmt das für ihn [25].

Für den Start der Anwendung ist es wichtig, dass eine *Activity* als Haupt-*Activity* gekennzeichnet wird, die beim Start der Anwendung ausgeführt wird. Diese Kennzeichnung wird in der Manifest-Datei vorgenommen. Die Manifest-Datei ist die zentrale Beschreibungsdatei einer Anwendung. Sie enthält Informationen über die Bestandteile der Anwendung, sowie alle Berechtigungen und Hardwareanforderungen die von der Anwendung benötigt werden. Außerdem gibt sie Auskunft über die mindestens nötige *Android*-Version, die von der Anwendung gewünscht ist [25, 8].

Wird eine *Activity* aufgerufen, durchläuft sie einen sogenannten Lebenszyklus. Sie besitzt dabei zu jedem Zeitpunkt einen Zustand und kann mit anderen Komponenten kommunizieren. Es gibt zwei Methoden, die eine Unterklasse von *Activity* immer implementiert. Das sind die Methoden `onCreate(Bundle)` und `onPause()`. In der Methode `onCreate(Bundle)` wird eine *Activity* mit ihrer Benutzeroberfläche und dessen Komponenten initialisiert. Wenn der Benutzer beispielsweise durch eine Aktion eine neue *Activity* startet, wird in der aktuellen *Activity* die `onPause()` Methode aufgerufen. Der Aufruf dieser Methode ermöglicht es, alle wichtigen Änderungen des Benutzers zu speichern, sodass der Benutzer bei einem erneuten Aufruf, diese wieder genauso vorfindet [8].

Activities werden, in der Reihenfolge in der sie geöffnet werden, im sogenannten *Backstack* abgelegt. Dabei ist der Startbildschirm eines Gerätes der Ausgangspunkt. Durch das klicken auf das *App-Icon* wird eine Anwendung gestartet und die Haupt-*Activity* geladen. Startet diese *Activity* eine weitere *Activity*, so wird die Haupt-*Activity* gestoppt und im *Backstack* abgelegt. Wenn der Benutzer die Zurück-Taste betätigt wird die aktuelle *Activity* in den *Backstack* verschoben und die Haupt-*Activity* erscheint [25, 8].

2.6.2 Intent

In *Android* wird zum Nachrichtenaustausch die Klasse *Intent* verwendet. Ein *Intent* ist eine passive Datenstruktur die eine abstrakte Beschreibung einer auszuführenden Operation enthält oder über eine *Action*, die aufgetreten ist, informiert. *Intents* werden nicht nur zum Datenaustausch zwischen *Activities* eingesetzt, sondern auch zwischen anderen Komponenten wie beispielsweise einzelnen *Services* und *Broadcast Receiver*, auf die in den nachfolgenden Abschnitten genauer eingegangen wird. Ein *Intent* verwaltet intern ein sogenanntes *Bundle* zur Speicherung von Daten mittels einfacher Datentypen. Durch die Methode `Intent.putExtra(String key, String value)` können Daten in Form von Name-Wert-Paaren über den *Intent* an andere Komponenten übertragen werden. Es gibt drei Arten von Informationen, die zur Auflösung eines *Intent* verwendet werden. Das sind *Action*, *Type* und *Categorie*. Wird als *Action* die Zielkomponente mit ihrem Komponentennamen benannt, spricht man von einem expliziten *Intent*. Dieser wird direkt an die entsprechende Zielkomponente weitergeleitet. Implizite *Intents* benennen kein Ziel, sondern überlassen die Interpretation dem Betriebssystem. In diesem Fall ermittelt das Betriebssystem anhand des *Type* und der *Categorie* des *Intent*, die für die Bearbeitung am besten geeignete Komponente. Der *Type* beschreibt einen expliziten Typ bzw. *MIME-Type*, welcher aus den *Intent*-Daten erschlossen wird. Die *Categorie* eines *Intents* hingegen legt fest, an welche Arten von Komponenten der *Intent* übertragen wird. Sie werden normalerweise dazu verwendet um Komponenten anderer Anwendungen zu öffnen. Umgesetzt wird das mit Hilfe der Klassen *Broadcast Receiver* und *IntentFilter*, die in den folgenden Abschnitten beschrieben werden [11].

2.6.3 Intentfilter

Mit der Klasse *IntentFilter* können sich einzelne Komponenten einer Anwendung auf implizite *Intents* registrieren. Damit kann eine Anwendung festlegen, auf welche impliziten *Intents* sie reagiert und welche sie ignoriert. Diese werden in der Manifest-Datei eingetragen. Als Indikator können *Actions*, *Categories*, allgemeine Datentypen oder Schemas verwendet werden [11].

2.6.4 Broadcast Receiver

Der *Broadcast Receiver* ist eine Komponente, die auf systemweit versendete *Intents* reagiert. Mit Hilfe der Klasse *IntentFilter* kann der *Broadcast Receiver* diese *Intents* anhand der *Categorie*, *Action* oder *Type* filtern und damit auf ausgewählte *Intents* reagieren. Das *Android* System versendet wichtige Systeminformationen ebenfalls über *Intents*. Beispielsweise informiert das System über einen niedrigen Batteriestand oder einen abgeschlossenen Datentransfer. Implementiert eine Anwendung einen *Broadcast Receiver* der auf diese *Intents* anspringt, so kann sie entsprechende Maßnahmen einleiten. Hierfür implementiert ein *Broadcast Receiver* die Methode `onReceive(Context context, Intent intent)`. Diese Methode wird bei jedem *Intent* aufgerufen, den der *Broadcast Receiver* empfängt. Innerhalb dieser Methode, kann dann auf verschiedene *Actions* abgefragt werden [25, 9].

2.6.5 Fragment

In der *Android* Version 3.0 hat *Google Fragments*, zur Verbesserung der Benutzeroberfläche auf *Tablets*, eingeführt. Seit der Version 4.0 *Ice Cream Sandwich* steht dies Komponente auch für *Smartphones* zur Verfügung. Die Trennung von Übersicht und Detaildarstellung, wie sie bei *Smartphones* nötig ist, ist bei *Tablets* aufgrund ihrer Größe nicht erforderlich. Beispielsweise bietet eine Chatanwendung auf einem *Smartphone* zuerst eine Liste aller Chaträume als Übersicht an. Wenn der Benutzer einen Chat auswählt, wird eine weitere *Activity* mit dem Chatverlauf geöffnet. Auf Geräten mit großen Displays ist dies völlig unnötig. Hier kann die Übersicht und die Detailansicht nebeneinander dargestellt werden, was dem Benutzer erlaubt Funktionen auszuwählen, ohne das derzeitig ausgeführte Modul verlassen zu müssen. Die Abbildung 2.2 zeigt den Aufbau einer Chatanwendung auf einem *Smartphone* und einem *Tablet*. Dabei sind *Fragments* immer an eine *Activity* gebunden und sind an dessen Lebenszyklus gebunden. Zudem können sie ebenfalls auf die *Backstack* Funktionalität, wie sie von *Activities* bekannt sind, zurückgreifen [25, 10].

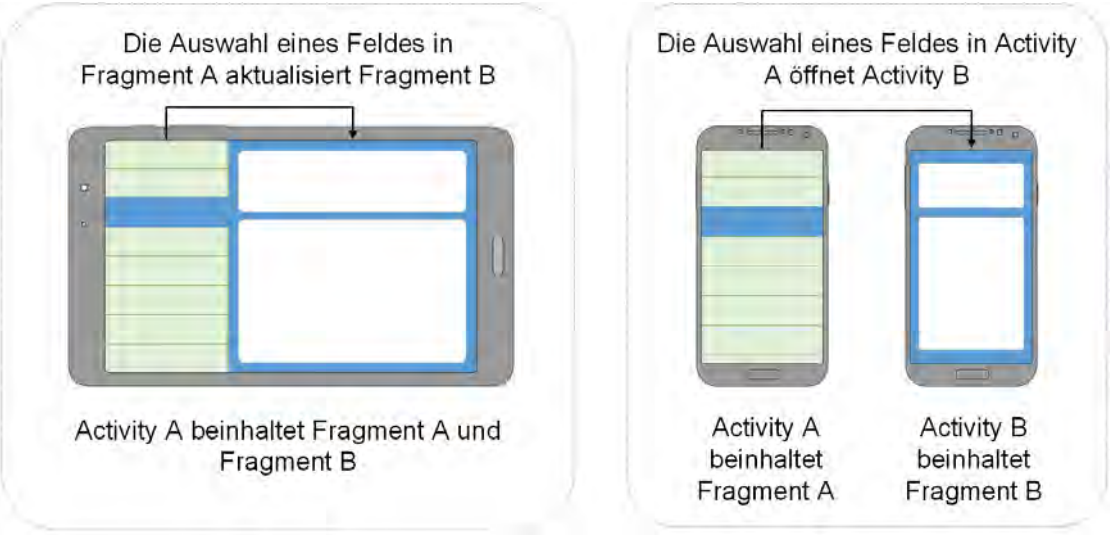

Abbildung 2.2: Einsatz von *Fragments* auf einem *Smartphone* und *Tablet* [10]

2.6.6 Service

Activities sind durch ihre Benutzeroberfläche für den Anwender unmittelbar sichtbar. Wählt der Benutzer in einer Radio-Anwendung einen Sender aus und startet die Übertragung, ist anschließend die Benutzeroberfläche dieser Anwendung für den Benutzer uninteressant. Der Benutzer verlässt also die Anwendung um einer anderen Aktivität nachzugehen. Damit der Radiosender trotzdem weiter gesendet wird, muss das System den Stream im Hintergrund weiter ausführen. Zu diesem Zweck stellt *Android* die Klasse *Service* bereit. *Services* haben keine Benutzeroberfläche und arbeiten im Hintergrund [25, 12].

Im Gegensatz zu *Broadcast Receivern* werden *Services* nicht nur beim Eintreten einer *Action* aktiviert, sondern können über die Methode `startService()` gestartet werden. Dies ist zu empfehlen, wenn der *Service* nur einmal gestartet werden muss und kein Ergebnis zurück erwartet wird. Soll zum Beispiel eine Datei mittels eines *Service* übertragen werden, macht es Sinn den *Service* an eine *Activity* zu binden. Dies kann über die Methode `bindService(Intent intent, ServiceConnection sConnection, int flags)` geschehen. Das ermöglicht es beispielsweise, die *Activity* zu einem späteren Zeitpunkt über den erfolgreichen Abschluss der Übertragung zu informieren. Im

Gegensatz zu *Broadcast Receivern* ermöglichen Services, den Einsatz bei länger andauernden Operationen [25, 12].

Im folgenden Abschnitt 2.7 wird der für diese Masterarbeit wichtige *Android* Ableger *Android Wear* vorgestellt.

2.7 Android Wear

Mit dem *Android* Ableger *Android Wear* verfolgt *Google* das Ziel, eine Plattform für *Wearables* zu gestalten. Dabei soll der Benutzer sein *Smartphone* in der Tasche behalten und mit einem Blick auf sein Handgelenk über Benachrichtigungen informiert werden. Dem Nutzer soll es einfach möglich sein, eingehende Benachrichtigungen zu erhalten und umgehend mit der Außenwelt weiter zu kommunizieren [3].

Aufgrund der geringen Größe von *Wearables* ist unter *Android Wear* ein erweitertes Konzept für die Bedienbarkeit von Anwendungen notwendig. In den nachfolgenden Abschnitten sind hierzu neue Konzepte und Komponenten beschrieben.

2.7.1 Pages

Die Benutzeroberfläche von *Android Wear* besteht im Wesentlichen aus sogenannten *Pages*. Diese *Pages* enthalten die darzustellenden Informationen in Form von *Cards* oder *List*, welche in den nachfolgenden Abschnitten genauer erläutert werden. *Pages* können vertikal und horizontal angeordnet sein. Mittels Wischgesten kann der Benutzer zwischen einzelnen *Pages* interagieren [18].

2.7.2 Cards

Eine *Card* stellt eine Komponente dar, die Informationen für den Benutzer in einer einheitlichen Art und Weise präsentieren kann. Hierfür stellt die *Wearable UI Library* unterschiedliche Implementierungen für *Android Wear*- Geräte zur Verfügung. Dazu zählt

die Klasse *CardFrame*, die eine *View* in einen kartenähnlichen Rahmen mit weißem Hintergrund und abgerundeten Ecken hüllt. Eine *Card* kann nur ein direktes Kind-Element enthalten. Normalerweise wird dort ein *Layout Manager* eingesetzt, der wiederum weitere Komponenten verwalten und somit den Inhalt einer *Card* steuern kann [14].

Es gibt zwei Möglichkeiten solche *Cards* in einer Anwendung zu verwenden. Ein *CardFrame* kann direkt in eine Layout-Datei aufgenommen werden. Dieser Ansatz bietet sich an wenn der Inhalt einer *Card* benutzerspezifisch festgelegt werden soll. Die *Wearable UI Library* bietet aber auch eine standardisierte *Card* über die Klasse *CardFragment* an. Diese *Card* verfügt über drei Komponenten, einem Titel, einer Beschreibung und einem Symbol. Die Abbildung 2.3 zeigt ein *CardFragment* mit diesen Komponenten. Der blaue Punkt stellt in diesem Fall einen Platzhalter für ein Symbol dar. Das *CardFragment* empfiehlt sich, wenn diese drei Komponenten die Anforderungen an die *Card* ausreichend erfüllen [14].

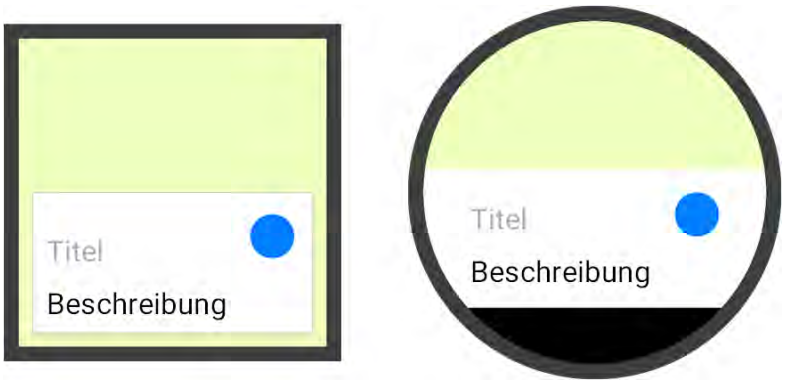

Abbildung 2.3: Das Standard *CardFragment* Layout [14]

2.7.3 List

Die *Wearable UI Library* verfügt mit der Klasse *WearableListView* über eine für *Smartwatches* optimierte Implementierung einer *List*. Mit einer *List* ist es dem Benutzer möglich, aus mehreren verschiedenen Elementen eines auszuwählen. Eine *List* kann mit Hilfe des *WearbleListView*-Elements in einer Layout-Datei deklariert werden. Anschließend kann eine eigene Layout-Datei für die einzelnen Listenelemente erstellt werden.

Bei der Implementierung einer Anwendung, sollte darauf geachtet werden, dass unter *Android Wear* eine *List* maximal drei Listenelemente auf dem Display sichtbar darstellen kann. Dazu teilt die Klasse *WearableListView* das Display in drei Bereiche. In Abbildung 2.4 a) sind diese Bereiche als Listenelement 1-3 benannt. Sie können weder von Entwicklern noch von Benutzern vergrößert oder verkleinert werden. Dadurch ist auch das Layout für einzelne Listenelemente räumlich eingeschränkt. Deshalb sollten Entwickler bei der Implementierung ihrer Anwendung darauf achten, keine großen Komponenten oder lange Texte in einer *List* darzustellen. Wie viele Komponenten verwendet werden können, hängt allein von dessen Größe und Inhalt ab [16]. Die Abbildung 2.4 zeigt, wie eine *List* unter *Android Wear* aussehen kann.

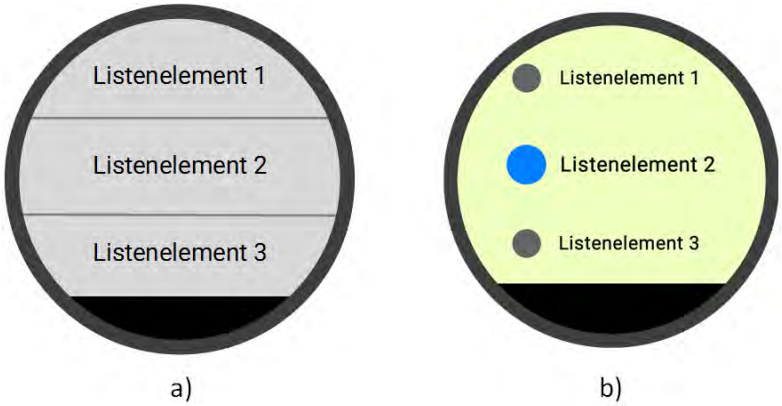

Abbildung 2.4: a) Bereiche für Listenelemente in *Android Wear* b) Darstellung einer *List* in *Android Wear* unter Verwendung der Klasse *WearableListView* [16]

2.7.4 2D Picker Pattern

Das *2D Picker Pattern* wird in *Android Wear* eingesetzt, um zwischen einzelnen *Pages* zu navigieren. Dieses Pattern lässt sich über die Klasse *GridViewPager* aus der *Wearable UI Library* realisieren. Sie implementiert einen sogenannten *Layout Manager* der den Benutzer in die Lage versetzt, mittels Wischgesten zwischen vertikal und horizontal angeordneten *Pages* zu interagieren [13].

Um einen *GridViewPager* mit mehreren *Pages* zu befüllen kommt die Klasse *FragmentGridPagerAdapter* zum Einsatz. Sie verwaltet ein oder mehrere *Fragments* und bestimmt dessen Position im *GridViewPager*. Das Nutzungskonzept von *Android Wear* sieht vor, dass horizontal angeordnete *Pages* in ihrer Anordnung nach rechts immer detailliertere Informationen zur vorhergehenden *Page* beinhalten. Beispielsweise kann in einer Wetter Anwendung auf der ersten *Page* die aktuelle Temperatur und die derzeitige Wetterlage angezeigt werden. Mit einer Wischgeste nach links, wird eine weitere *Page* sichtbar, welche die aktuelle Wetterlage mit Temperatur, Windstärke, Luftfeuchtigkeit und Regenrisiko präsentiert [13].

2.7.5 WatchViewStub

Unter *Android Wear* werden die gleichen Layout Techniken wie bei *Android* verwendet. Es müssen nur bestimmte Einschränkungen beachtet werden. Anders als bei *Android* unterstützt *Android Wear* runde und rechteckige Bildschirmformen [15].

Mit der Klasse *WatchViewStub* aus der *Wearable UI Library* können verschiedene Layouts für diese definiert werden. Die Klasse ermittelt zur Laufzeit die Form des verwendeten Bildschirms und lädt diese entsprechend. Um diese Klasse zur Verwaltung von verschiedenen Bildschirmformen in einer Anwendung einsetzen zu können, muss das *WatchViewStub-Tag* in die Layout-Datei der *Activity* eingefügt werden. Innerhalb des *WatchViewStub-Tag* kann mit den Attributen `app:rectLayout` und `app:roundLayout` unterschiedliche Layout-Dateien für runde und rechteckige Bildschirmformen referenziert werden. Dabei werden die Layout-Dateien unter *Android Wear* auf die gleiche Art und Weise wie in *Android* erzeugt [15].

Anschließend kann in der `onCreate(Bundle bundle)` Methode der *Activity* die Layout-Datei, die den *WatchViewStub* enthält, über die Methode `setContentView(int layout\_id)` geladen werden. Nachfolgend ist dieser Vorgang in Abbildung 2.5 nochmals graphisch dargestellt.

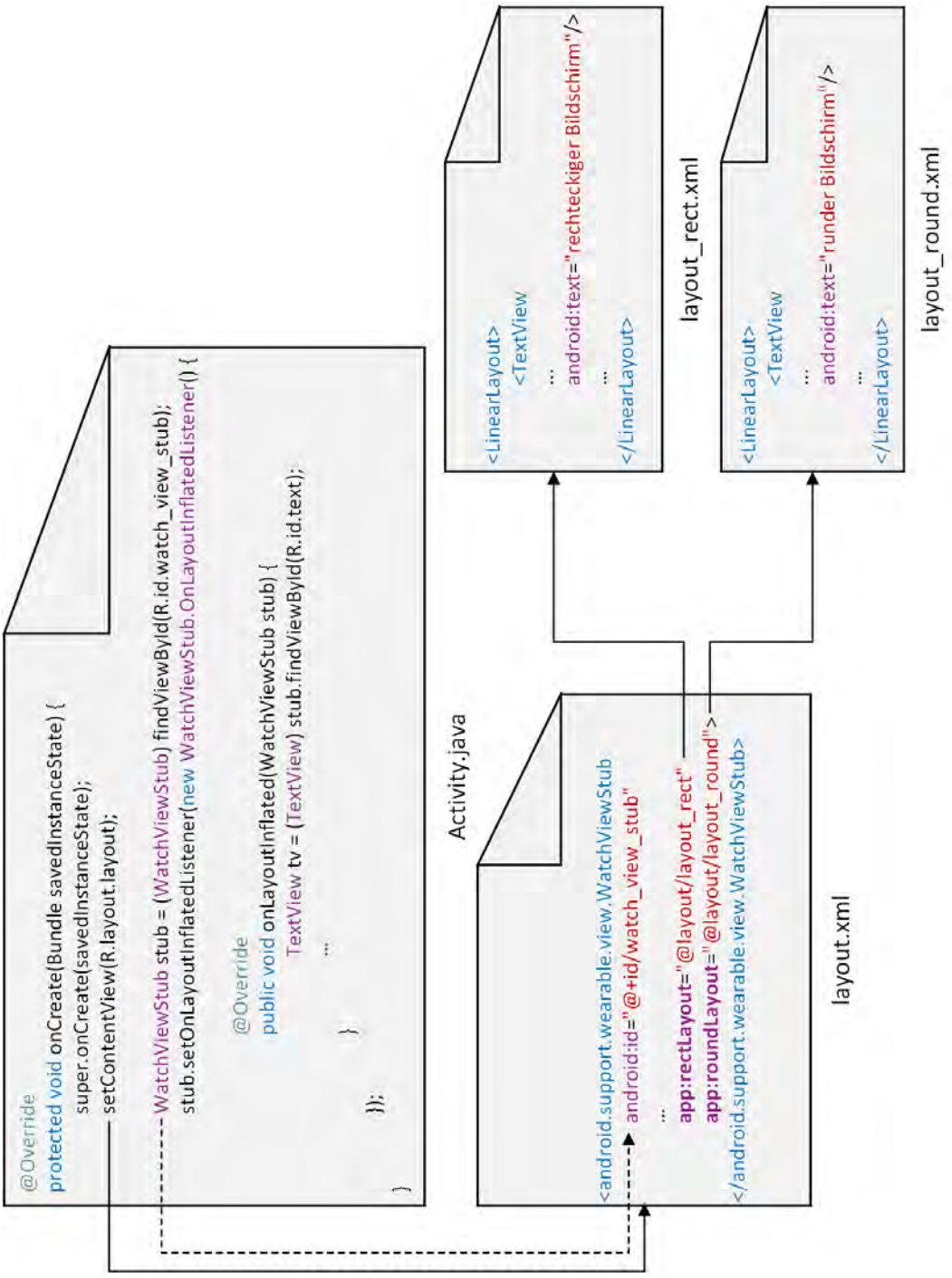

Abbildung 2.5: Laden von unterschiedlichen Layout-Dateien für unterschiedliche Bildschirmformen

2.7.6 BoxInsetLayout

Die Klasse *BoxInsetLayout* bietet eine Alternative zu der im letzten Abschnitt beschriebenen Klasse *WatchViewStub*. Sie ist ebenfalls Bestandteil der *Wearble UI Library* und erweitert das aus *Android* bekannte *FrameLayout*. Mit dieser Klasse ist es möglich, eine Layout-Datei für runde und rechteckige Bildschirmformen zu erstellen. Dafür verwendet die Klasse einen Fenstereinsatz abhängig von der verwendeten Bildschirmform. Dieser Fenstereinsatz wird wie in Abbildung 2.6 zu sehen, je nach Bildschirmform anders dargestellt. Das graue Rechteck in Abbildung 2.6 zeigt den Bereich des *BoxInsetLayout*, in denen auf runden Bildschirmen deren Kindelemente automatisch angeordnet werden. Die Breite des schwarzen Randes bei runden Bildschirmen wird mittels des `layout\_ box` Attributes festgelegt. Auf rechteckigen Bildschirmen wird dieses Attribut ignoriert und das *BoxInsetLayout* füllt den Bildschirm vollständig aus [15].

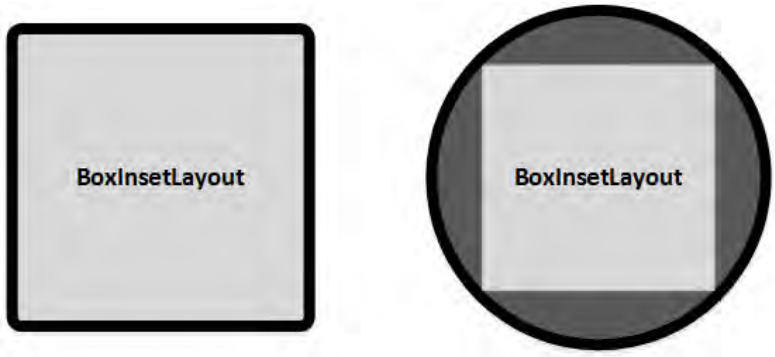

Abbildung 2.6: *BoxInsetLayout* auf einem rundem und einem rechteckigen Bildschirm [15]

2.7.7 Wearble Data Layer API

Die *Wearable Data Layer API* ist Teil der *Google Play Services*. Sie stellt einen Kommunikationskanal für *Android* und *Android Wear*- Anwendungen bereit. Die API besteht aus einer Reihe von Datenobjekten, die das System über eine bestehende Verbindung senden und synchronisieren kann. Anwendungen können sich mittels eines *Service* auf wichtige Ereignisse im *Data Layer* registrieren [17].

Zur Kommunikation zwischen *Android*- und *Android Wear*- Anwendung bietet die *Wearable Data Layer API* zwei verschiedene Klassen, die *Wearable.MessageApi* und *Wearable.DataApi*, an. Diese werden in den nachfolgendenden Abschnitten einzeln beschrieben [17].

Wearable.MessageApi

Die Klasse *Wearable.MessageApi* kann zum Versenden von einzelnen *Messages* verwendet werden und eignet sich besonders gut für *Remote Procedure Calls (RPC)* oder in *Request/Response* Kommunikationsmodellen für einzelne Anfragen. Beispielsweise kann mit diesen *Messages* eine entfernte Steuerung eines Media Players auf dem *Smartphone*, durch eine *Android Wear*- Anwendung realisiert werden [17, 21].

Ist eine *Smartwatch* mit einem *Smartphone* verbunden, puffert das System diese *Messages* in einer Warteschlange und informiert den Sender über die erfolgreiche Übertragung. Ist kein Empfänger vorhanden, wird ein Fehler zurückgeliefert. Die Signalisierung einer erfolgreichen Übertragung bedeutet jedoch nicht, dass die *Message* auch wirklich beim Empfänger angekommen ist, da ein Verbindungsabbruch vor dem Auslesen der *Message* aus der Warteschlange immer noch möglich ist [17, 21].

Wearable.DataApi

Mit der Klasse *Wearable.DataApi* bietet *Google* eine weitere Datenstruktur für die Kommunikation zwischen *Android*- und *Android Wear*- Anwendung an. Mit ihr können Daten zwischen *Smartphone* und *Smartwatch* synchronisiert werden. Zu diesem Zweck nutzt sie die Klasse *DataItem*. Sie definiert die Schnittstelle zwischen dem System und den zu synchronisierenden Daten. Ein *DataItem* besteht typischerweise aus einem Pfad - *Path* - und den eigentlichen Daten - *Payload*. Dabei definiert der *Path* eine eindeutige Zeichenkette die das *DataItem* identifiziert. Die *Payload* hingegen enthält die eigentlichen Daten. Es muss allerdings beachtet werden, dass die enthaltenen Daten nicht größer als 100 KB sein dürfen. Sonst wird bei der Übertragung durch die *Wearable.DataApi* ein Fehler signalisiert und die Synchronisierung abgebrochen [17, 19].

Anwendungen können sich über Änderungen von einzelnen *DataItems* von der *Wearable Data Layer API* benachrichtigen lassen. Hierzu wird die Klasse *WearableListenerService* eingesetzt, die von der *Android* Klasse *Service* ableitet. Dieser *Service* ermöglicht es, sich auf einzelne Ereignisse der *Wearable Data Layer API*, wie beispielsweise einer Änderung der *Payload* eines *DataItems*, zu registrieren [19].

Kommunikationsaufbau

Mit *Android Wear* ist es möglich, mehrere *Wearable* mit einem *Smartphone* zu verwenden. Dafür setzt *Google* auf verschiedene Verbindungen zwischen den einzelnen Geräten. Zwischen *Smartphone* und *Smartwatch* wird als Übertragungsmedium *Bluetooth LE* verwendet. Seit der Version 5.1.1 wird auch *WLAN-Sync* unterstützt[1]. Die Abbildung 2.7 zeigt die einzelnen Kommunikationspfade auf. Um Daten zwischen einem *Smartphone* und mehreren *Wearables* zu synchronisieren stellt *Google* eine *Cloud Node* auf ihren Servern bereit. Alle übertragenen Daten werden mit der *Cloud Node* synchronisiert. Bei Änderungen werden alle direkt mit der *Cloud Node* verbundenen Geräte informiert und synchronisiert [17].

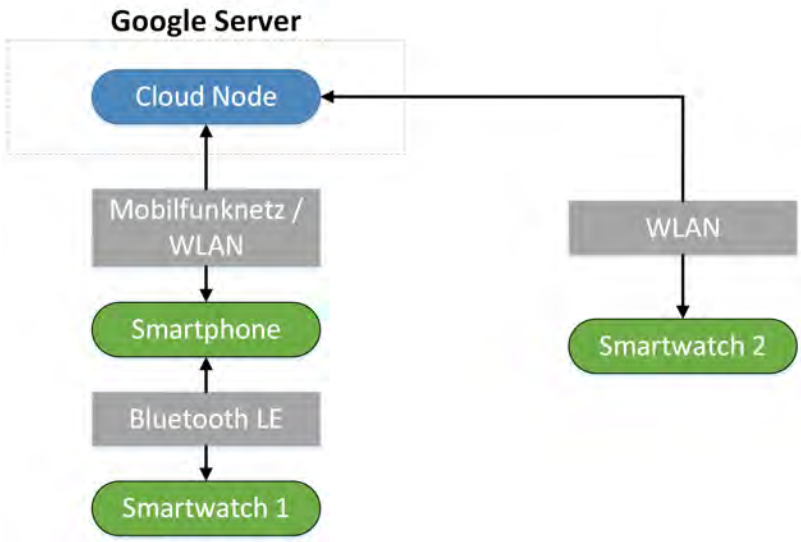

Abbildung 2.7: Mögliche Kommunikationswege zwischen einem *Smartphone* und einer *Smartwatch* mit *Android Wear* Betriebssystem [17]

2.7.8 Packaging Mechanismus

Anders als das normale *Android* Betriebssystem hat *Android Wear* keinen direkten Zugang zum *Google Play Store* und kann somit Anwendungen nicht direkt herunterladen und installieren. Zu diesem Zweck hat *Google* einen Mechanismus mit dem Namen *Packaging* eingeführt [20].

Eine *Android Wear*- Anwendung wird von Entwicklern immer mit der eigentlichen *Smartphone* Anwendung ausgeliefert. Das heißt, dass mit dem Herunterladen einer Anwendung aus dem *Play Store*, die *Android Wear*- Anwendung automatisch mit heruntergeladen wird. *Google* sieht bei *Android Wear*- Geräten immer eine Verbindung zu einem *Smartphone* vor. Um ein *Android Wear*- Gerät mit einem *Smartphone* verbinden zu können, muss das *Smartphone Android* 4.3 (API Level 18) oder eine neuere Version von Android verwenden. Für die Kommunikation zwischen *Smartphone* und *Smartwatch*, wird die von *Google* bereitgestellte *Android Wear Companion*- Anwendung aus dem *Play Store* benötigt. Diese erkennt, welche installierte Anwendung eine *Android Wear*- Anwendung kapselt und installiert diese auf der *Smartwatch* [20].

Damit dieser Vorgang reibungslos funktioniert, müssen Entwickler auf drei Dinge achten. Es muss beachtet werden, dass die *Android*- und *Android Wear*- Anwendung über die gleichen Berechtigungen und Versionsnummern verfügen. Außerdem müssen sie das gleiche *Package* implementieren. Zuletzt muss die *Android Wear*- Anwendung als Abhängigkeit in die *Android* Anwendung aufgenommen werden. Nur so wird die *Android Wear*- Anwendung zusammen mit der *Android* Anwendung erstellt [20].

3

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die zu implementierenden Anwendungsprototypen analysiert. Hierfür werden zu Beginn, die für diese Anwendung definierten Anwendungsfälle betrachtet. In der Folge werden daraus die Anforderungen an die Anwendung definiert und in funktionale und nicht funktionale Anforderungen unterteilt.

3.1 Anwendungsfälle

Ein Nutzer erwartet von einer Umfragesystem, dass er mit dieser bequem an Umfragen teilnehmen kann. Dazu sollen die Anwendungen dem Nutzer einen Überblick über zur Verfügung stehende Umfragen anzeigen und ihm gleichzeitig die Möglichkeit der Bearbeitung ermöglichen. Außerdem soll der Nutzer über neue Umfragen umgehend benachrichtigt werden können.

Umfragen werden in Form von einzelnen Fragebögen an den Nutzer ausgeliefert. Fragebögen können sich in vielerlei Hinsicht unterscheiden. Sie können eine unterschiedliche Anzahl an Fragen enthalten und verschiedene Fragearten, wie beispielsweise *Multiple Choice* und *Single Choice*, verwenden. Auch die Fragen selbst können unterschiedliche Eigenschaften im Hinblick auf die Länge des Fragetextes oder der Anzahl an Antwortoptionen aufweisen. Auch die Textlänge der Antwortoptionen erscheint, aufgrund der technischen Einschränkungen, im Hinblick auf *Smartwatches* wichtig. Diese Eigenschaften müssen bei der Entwicklung des Umfragesystems beachtet werden. Es müssen die unterstützten Fragearten festgelegt und dafür notwendigen Anforderungen definiert werden. Die Tabelle 3.1 listet die Fragearten auf, die das Umfragesystem unterstützen

soll. Dabei soll es auf der *Smartwatch* und dem *Smartphone* möglich sein, Fragebögen vollständig zu beantworten und zu speichern. Außerdem soll darauf geachtet werden, dass Fragen und deren Antwortoptionen vollständig dargestellt werden können.

Frageart	**Beschreibung**
Multiple Choice	Mehrere Antwortoptionen können angekreuzt werden.
Single Choice	Exakt eine Antwortoption kann angekreuzt werden.
Skalen	Es kann ein Wert auf einer Skala selektiert werden
Offene Fragen	Es kann eine Antwort über Texteingabe gegeben werden.

Tabelle 3.1: Unterstützte Fragearten

Das zu entwickelnde Umfragesystem soll in erster Linie zur Durchführung einer Studie herangezogen werden. In den Abschnitten 2.7.2 und 2.7.3 wurden die unter *Android Wear* zur Verfügung stehenden Darstellungsformen *List* und *Cards* vorgestellt. In dieser Masterarbeit soll evaluiert werden, welche dieser Darstellungsformen für die Beantwortung von *Multiple Choice* und *Single Choice* Fragen auf *Smartwatches* am sinnvollsten ist. Außerdem soll evaluiert werden, ob sich die Spracheingabe zur Beantwortung von offenen Fragen auf der *Smartwatch* eignet. Weiterhin soll evaluiert werden, ob Fragen mittels einer Skala auf der *Smartwatch* beantwortet werden können.

Zur Erleichterung der Erfassung und Speicherung von Zeitmessungen und Benutzerinteraktionen auf dem *Smartphone* als auch auf der *Smartwatch*, soll das Umfragesystem entsprechend erweitert werden. Hierzu müssen weitere funktionale Anforderungen an die Anwendung für *Smartphone* und *Smartwatch* definiert werden.

In den nachfolgenden Abschnitten werden aus den hier beschriebenen Anwendungsfällen Anforderungen abgeleitet und in funktionale und nicht funktionale Anforderungen unterteilt.

3.2 Funktionale Anforderungen

Die nachfolgende Tabelle 3.2 zeigt die funktionalen Anforderungen an das zu entwickelnde Umfragesystem.

Nr.	Bezeichnung	Beschreibung
***Smartphone*- und *Smartwatch*- Anwendung**		
1.	Fragebögen anzeigen	Die Anwendung soll alle zur Verfügung stehenden Fragebögen anzeigen.
2.	Beantworten von Fragebögen	Fragebögen sollen auf *Smartphone* und *Smartwatch* beantwortet werden können.
3.	Benachrichtigung	Der Nutzer soll auf *Smartphone* und *Smartwatch* über neue Umfragen benachrichtigt werden können.
4.	Unterstützung von Fragearten	Die Anwendungen sollen die in der Tabelle 3.1 definierten Fragearten unterstützen.
***Smartphone*- Anwendung**		
5.	*Smartphone* Zeitmessung	Die Anwendung soll die Bearbeitungszeit für einen ganzen Fragebogen aufzeichnen.
6.	Texteingabegeschwindigkeit	Die Anwendung soll die Eingabegeschwindigkeit bei offenen Fragen aufzeichnen.
***Smartwatch*- Anwendung**		
7.	Delegieren von Fragebögen an *Smartphone*	Der Nutzer soll während der Bearbeitung eines Fragebogens, diesen an das *Smartphone* delegieren können, um dort mit der Bearbeitung fortzufahren.
8.	Darstellungsformen	Die Anwendung soll für *Single Choice*- und *Multiple Choice*- Fragen die beiden Darstellungsformen *List* und *Cards* erlauben.

Fortsetzung auf Folgeseite

Tabelle 3.2: Funktionale Anforderungen

Nr.	Bezeichnung	Beschreibung
9.	Spracherkennung	Offene Fragen sollen mit der Spracherkennung beantwortet werden können.
10.	Fragen mit Skalen	Fragen die eine Skala beinhalten, sollen beantwortet werden können.
11.	*Smartwatch*-Zeitmessung	Die Anwendung soll die Bearbeitungszeit einzelner Fragen sowie des gesamten Fragebogens aufzeichnen.
12.	Aufzeichnung von Benutzerinteraktionen	Die Anwendung soll die Aufzeichnung von Benutzerinteraktionen erlauben.
13.	Spracheingabe-Zeitmessung	Die Anwendung soll die Zeit aufzeichnen, die der Nutzer für die Spracheingabe benötigt.

Tabelle 3.2: Funktionale Anforderungen

3.3 Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen definieren die Anforderungen an das Umfragesystem im Hinblick auf Aussehen und Handhabung. Die nachfolgende Tabelle 3.3 zeigt die nicht funktionalen Anforderungen an das zu entwickelnde Umfragesystem.

Nr.	Bezeichnung	Beschreibung
***Smartphone*- und *Smartwatch*- Anwendung**		
1.	Einfache Bedienung	Die entwickelte App zielt auf eine einfache Interaktion ohne spezielle Fachkenntnisse.
2.	Minimale Verzögerung	Die entwickelte App besitzt ein Minimum an Verzögerung.

Tabelle 3.3: Nicht funktionale Anforderungen

4

Architektur

In Kapitel 4 wird die Architektur des entwickelten Umfragesystems vorgestellt. Kapitel 4 soll als Vorlage für die darauf folgende Implementierung des Umfragesystems dienen. Der Abschnitt 4.1 beschreibt den architektonischen Aufbau des Systems. In den darauf folgenden Abschnitten 4.2 und 4.3 wird auf die einzelnen Systemkomponenten eingegangen.

4.1 Systemaufbau

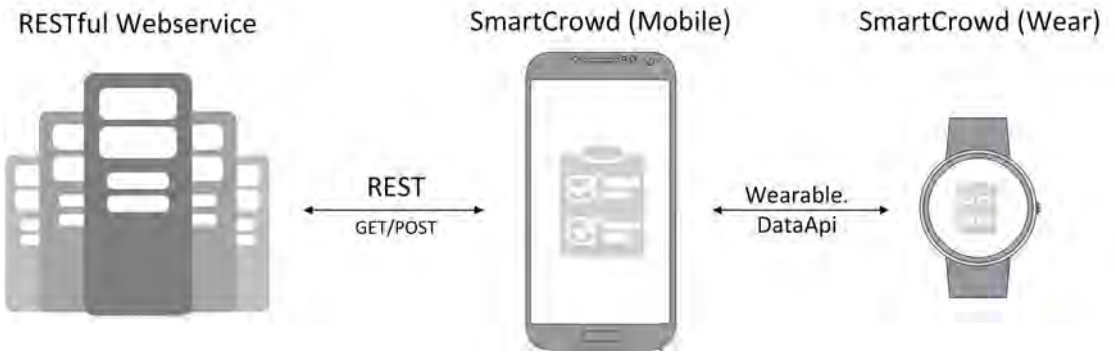

Abbildung 4.1: Systemarchitektur

Die Abbildung 4.1 zeigt den grundlegenden Aufbau des Umfragesystems mit den einzelnen Systemkomponenten. Es stellt die Basis für die Implementierung dar. Aufgrund der zur Verfügung stehenden Geräte wurde entschieden die Implementierung der mobilen Anwendungen auf Basis von *Android* und *Android Wear* durchzuführen.

4.2 Webservice

Der Webservice ist die zentrale Komponente des Umfragesystems und dient in erster Linie zur Bereitstellung von Umfragen in Form von einzelnen Fragebögen. Dies geschieht mittels Einsatz einer sogenannten *REST*-API. Diese API bietet eine standardisierte Schnittstelle zum Abrufen, Speichern und Löschen von Datensätzen. Durch den Einsatz einer solchen webbasierten Lösung ist die Erreichbarkeit des Umfragesystems über das Internet gewährleistet. Für die geplante Studie soll der Webservice zusätzlich über die Verwaltung von Log-Einträgen ausgestattet werden, die auf den Geräten aufgezeichnet werden.

4.3 Mobile Anwendungen

Eine direkte Verbindung zwischen einer *Smartwatch* auf Basis von *Android Wear* und einem entfernten Webservice wird zum Zeitpunkt dieser Arbeit nicht unterstützt. Zu diesem Zweck wird eine aktive Verbindung zu einem *Android Smartphone* benötigt. Das kann für die *Smartwatch* als *Proxy*[1] fungieren. Das *Smartphone* muss hierfür mit dem Betriebssystem *Android* 4.3 (API Level 18) oder höher ausgestattet sein.

Damit das Zusammenspiel von *Smartphone* und *Smartwatch* funktioniert, wird auf beiden Geräten jeweils eine Anwendung benötigt. Diese können später untereinander Informationen austauschen und zusammenarbeiten.

4.3.1 Anwendungskonzept

Die für das *Smartphone* entwickelte Anwendung wird in Abbildung 4.1 als *SmartCrowd (Mobile)* bezeichnet. Sie dient primär als *Proxy* zum Empfangen und Senden von Umfragen und Umfrageergebnissen durch die *Smartwatch*. Zu Studienzwecken bietet sie ebenfalls eine Benutzeroberfläche zur Bearbeitung von Umfragen.

[1] Ein *Proxy* ist ein Gerät in einem Netzwerk, dass Anfragen entgegennimmt und über seine eigene Adresse weiterleitet

Die *SmartCrowd (Wear)*- Anwendung auf der *Smartwatch* bietet die Möglichkeit Umfragen zu bearbeiten und im Anschluss die Umfrageergebnisse über das *Smartphone* an den Webservice zurück zu schicken. Mit der entwickelten Anwendung sollen außerdem verschiedene Eingabeformen für Fragen evaluiert und getestet werden. Für die geplante Studie sollen beide Anwendung mit einem Log-Mechanismus ausgestattet werden. Dieser soll später bei der Durchführung der Studie zur Zeitmessung und Benutzerinteraktionen eingesetzt werden können.

Smartphone- und *Smartwatch*- Anwendung werden in *Android* immer zusammen in einer Anwendung ausgeliefert. Die Abbildung 4.2 zeigt in einem Paketdiagramm den Aufbau dieser Anwendung. Die Pakete *SmartCrowd_ Mobile* und *SmartCrowd_ Wear* stellen die *SmartCrowd (Mobile)*- und *SmartCrowd (Wear)*- Anwendung dar. Das Paket *CrowdSensingLibrary* kapselt die Anwendungslogik für die Anbindung an den Webservice als Bibliothek. So kann diese später auch in weiteren Anwendungen verwendet werden. Auch die Kommunikationslogik zwischen *SmartCrowd (Mobile)* und *SmartCrowd (Wear)* wird in einer *Android* Bibliothek abstrahiert. Diese ist in Abbildung 4.2 als *WearConnectLibrary* gekennzeichnet.

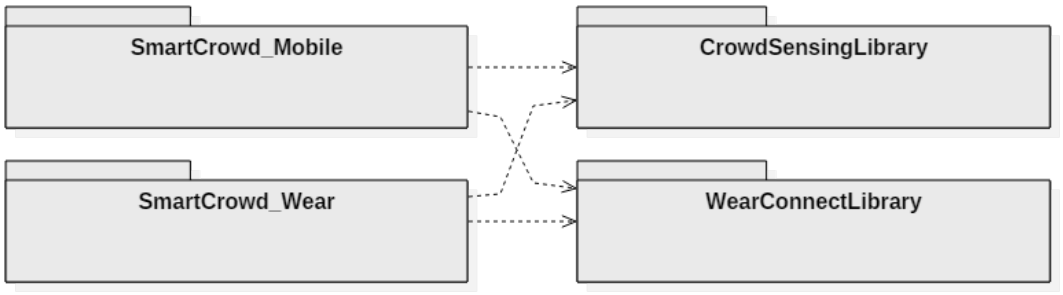

Abbildung 4.2: Das Anwendungskonzept als Paketdiagramm

4.3.2 Erweitertes Anwendungskonzept für Smartwatches

Smartwatches unterscheiden sich bei der Benutzerinteraktion maßgeblich mit der eines *Smartphones*. Wie im Kapitel 2 dargestellt, bieten *Smartwatches* auf Basis von *Android*

Wear im Wesentlichen zwei Darstellungsformen an. Das sind die sogenannten *Cards* und die *List*. In dieser Masterarbeit soll evaluiert werden, welche Darstellungsformen für die Beantwortung von Fragebögen am sinnvollsten ist. Außerdem soll evaluiert werden, ob die Beantwortung von Fragen mittels Skalen auf der *Smartwatch* geeignet ist. Hierfür wird das im vorigen Abschnitt erläuterte Anwendungskonzept erweitert. Es werden verschiedene Eingabeformen für die unterstützten Fragearten dem Anwendungskonzept hinzugefügt. Dabei basieren diese Eingabeformen auf den Darstellungsformen *Cards* und *List* sowie eigens implementierten Darstellungsformen für Skalen. Die Eingabeformen sollen später in einer Studie auf ihre Tauglichkeit innerhalb eines *Mobile Crowd Sensing Systems* überprüft werden. Die Abbildung 4.3 zeigt das erweiterte Konzept zur Evaluierung der Eingabeformen.

Im nachfolgenden Kapitel 5 wird auf die Implementierung der vorgestellten Systemkomponenten eingegangen. Dabei werden auch die einzelnen Eingabeformen genauer beschrieben.

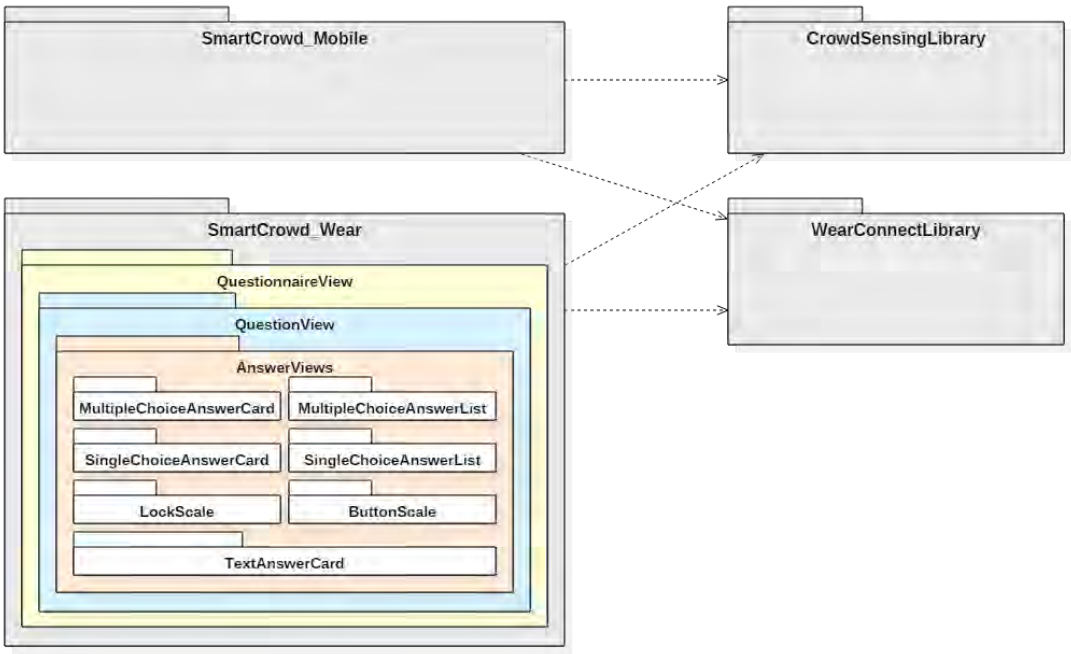

Abbildung 4.3: Erweitertes Anwendungskonzept mit verschiedenen Eingabeformen

5

Implementierung

In diesem Kapitel werden die im Kapitel 4 vorgestellten Softwarekomponenten in ihrer technischen Umsetzung beschrieben. Der Abschnitt 5.1 erläutert die generischen Bibliotheken. Die weiteren Abschnitte 5.2 bis 5.4 stellen die Anwendungen für den Webservice, das *Smartphone* und die *Smartwatch* vor. Abschließend werden in Abschnitt 5.5 die nötigen Anpassungen der Anwendungen für die geplante Studie beschrieben.

5.1 Bibliotheken

In den folgenden Abschnitten 5.1.1 bis 5.1.3 werden die generischen *Java*- und *Android* Bibliotheken beschreiben. Am Anfang wird die grundlegendste Bibliothek *CrowdSensingLibrary* vorgestellt. In Abschnitt 5.1.2 wird die *WearConnectLibrary* in ihrem Aufbau beschrieben. Zuletzt wird in Abschnitt 5.1.3 auf die, für die Studie erforderliche Bibliothek *LogLibrary* eingegangen.

5.1.1 CrowdSensingLibrary

Die *Java* Bibliothek *CrowdSensingLibrary* kapselt die Anwendungslogik, die sich mit der Verwaltung von Umfragen beschäftigt. Sie beinhaltet eine Struktur zur zentralen Verwaltung von Fragebögen sowie dessen Frage- und Antworttypen. Die dazu nötigen Pakete und Klassen sind in Abbildung 5.1 dargestellt.

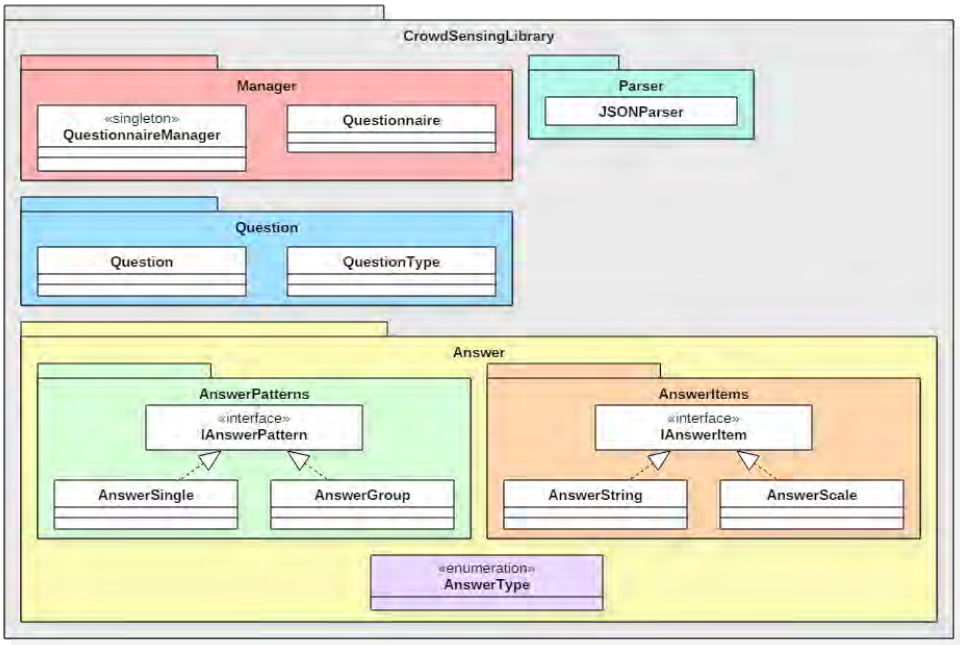

Abbildung 5.1: Klassendiagramm der *CrowdSensingLibrary* Bibliothek

Die *CrowdSensingLibrary* ist als *Singleton*[1] ausgelegt. Damit ist sichergestellt, dass innerhalb einer Anwendung nur eine Instanz der *CrowdSensingLibrary* verfügbar ist. Die Instanz der *CrowdSensingLibrary* kann über die Funktion `QuestionnaireManager.getInstance()` abgerufen werden.

Die Instanz des *QuestionnaireManager* bietet die Möglichkeit, eine Liste von Fragebögen zu verwalten. Dabei werden die in der Tabelle 3.1 aufgelisteten Fragearten *Multiple Choice*, *Single Choice*, Skalen und offene Fragen unterstützt. In der Klasse *QuestionType* sind diese definiert und können anhand dieser differenziert werden.

Über die Funktion `addQuestionnaire(Questionnaire q)` können der Instanz des *QuestionnaireManager* Fragebögen zur Verwaltung übergeben werden. Für Fragebögen, die im *JSON*-Datenformat zur Verfügung stehen, bietet der *QuestionnaireManager* zwei weitere Funktionen an. Mit der Funktion `addQuestionnaireJSON (String`

[1] Ein *Singleton* bezeichnet in der Softwareentwicklung ein Entwurfsmuster. Es stellt sicher, dass es von einer Klasse nur eine Instanz gibt [4].

`jsonString)` ist der *QuestionnaireManager* auch in der Lage Fragebögen im *JSON*-Datenformat aufzunehmen und mit der Funktion `String getQuestionnaireJSON()` diese als solche wieder herauszugeben. Da alle Fragebögen zwischen den einzelnen Anwendungen und dem Webservice im *JSON*-Datenformat ausgetauscht werden, unterstützen die genannten Funktionen diesen Prozess.

5.1.2 WearConnectLibrary

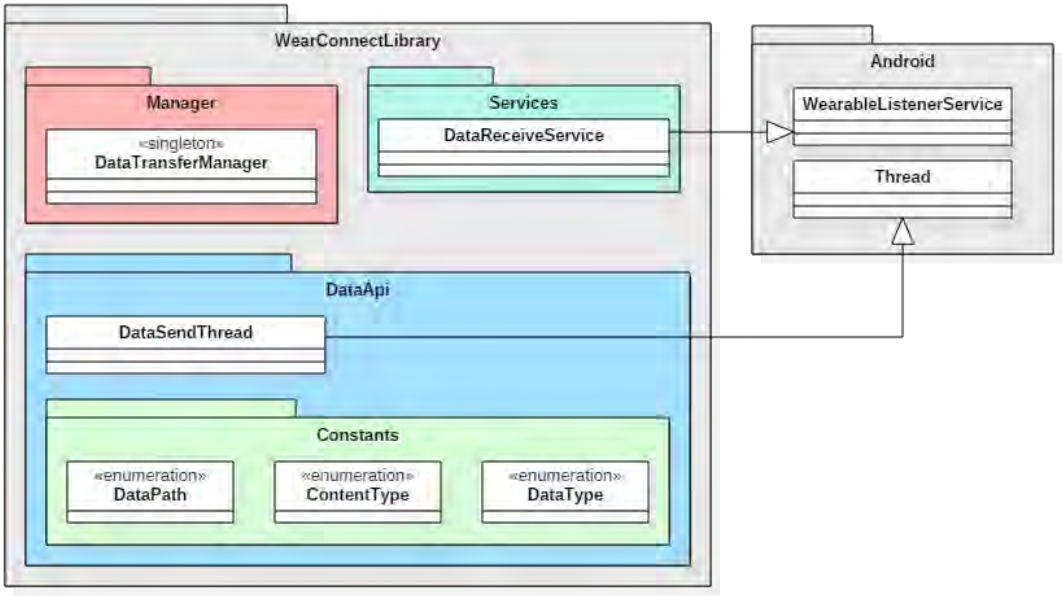

Abbildung 5.2: Klassendiagramm der *WearConnectLibrary* Bibliothek

Die *Android* Bibliothek *WearConnectLibrary* stellt eine Abstraktion der Kommunikationslogik zwischen der *Android*- und *Android Wear*- Anwendung dar. Zentrale Aufgabe der Bibliothek ist die Kommunikation zwischen den Anwendung sicherzustellen. Die dazu nötigen Pakete und Klassen sind in Abbildung 5.2 zu sehen.

Zum Datenaustausch zwischen einer *Android*- und *Android Wear*- Anwendung bietet *Google* die in Kapitel 2 ausführlich beschriebenen Bibliotheken *Wearable.DataApi* und *Wearable.MessageApi* an. Im Gegensatz zur *Wearable.DataApi* ist bei der *Wea-*

rable.MessageApi nicht sichergestellt, dass die gesendeten Daten beim Empfänger ankommen. In Kapitel 3 sind Anforderungen definiert, die diese Eigenschaft erforderlich machen. Aus diesem Grund nutzt die *WearConnectLibrary* die *Wearable.DataApi* zum Datenaustausch. Ein weiteres Ziel dieser Bibliothek ist die einfache Verwendung der *Wearable.DataApi* aus der Anwendung heraus. Bei der Entwicklung der Anwendungsschnittstelle wurde darauf geachtet, dass eine lose Anbindung an die *Wearable.DataApi* bereitgestellt wird. Zu diesem Zweck besitzt die Bibliothek die Klasse *WearTransferManager*. Sie stellt die statische Funktion `sendData(String dataPath, String contentType, String content)` bereit. Damit die Bibliothek mehrere Übertragungskanäle unterstützen kann, besitzt diese Funktion den Parameter *dataPath*, der den Namen des Übertragungskanals festlegt. Die beiden nachfolgenden Parameter definieren den Typ der zu übertragenden Daten und die Daten selbst. Die *WearConnectLibrary* stellt zusätzlich die Klassen *DataPath*, *DataType* und *ContentType* zur Verfügung. Sie enthalten vordefinierte statische *Strings*. Diese dienen zur Absicherung, sodass Sender als auch Empfänger die identischen Name-Wert-Paare für Nachrichtenpakete und Übertragungskanäle verwenden. Zum Empfangen von Nachrichtenpaketen, die über diese Bibliothek versendet werden, muss die entsprechende Anwendung den *DataReceiveService* implementieren. Die vererbte Methode `DataReceiveService.onDataChanged( DataEventBuffer dataEvent)` wird immer beim Erhalt einer Nachricht über die *Wearable.DataApi* aufgerufen. Da es unter *Android* nicht erlaubt ist, eine Netzwerkverbindung über den *UI-Thread* aufzubauen, wird ein neuer *Thread* mit dem Namen *DataSendThread* gestartet. Dort werden die Daten über die *Wearable.DataApi* übertragen. Wie aus dem Grundlagenkapitel 2.7.7 bekannt, hat die *Wearable.Data.Api* die Einschränkung, dass nur 100KB mittels eines *DataItems* übertragen werden können. Um diese Einschränkung zu umgehen, ist der *DataSendThread* mit einer *Splitting*-Methode versehen. Ist das zu übertragene Datenpaket größer als die 100KB wird das Datenpaket in mehrere Datenpakete geteilt und nacheinander in mehreren *DataItems* versendet. Eine beispielhafte Verwendung der *WearConnectLibrary* ist im Sequenzdiagramm 5.3 zu sehen.

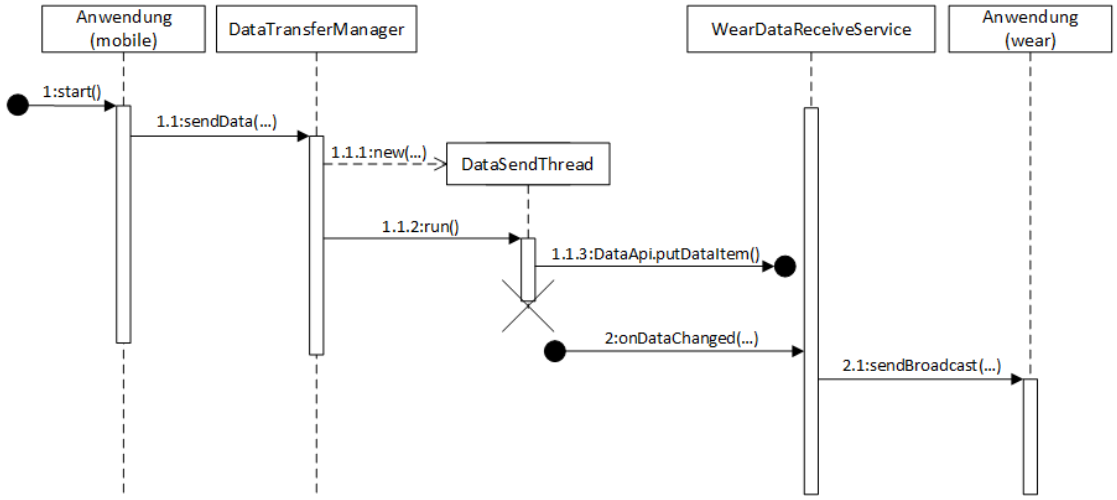

Abbildung 5.3: Sequenzdiagramm zur Verwendung der *WearConnectLibrary* Bibliothek

5.1.3 LogLibrary

Für die geplante Studie ist es erforderlich, Zeitmessungen innerhalb der Anwendungen vorzunehmen. Zusätzlich soll es möglich sein, Interaktionen des Benutzers aufzuzeichnen. Damit diese Funktionen in mehreren Anwendungen verwendet werden kann, wurde entschieden, diese Funktionalität in einer eignen Bibliothek zu kapseln. Der Log-Mechansimus soll vorwiegend im Hintergrund agieren und den Nutzer bei der Beantwortung des Fragebogens nicht beeinflussen.

Die Bibliothek stellt die beiden Strukturen *ActionLog* und *TimeLog* bereit. Dabei repräsentiert eine Instanz der Klasse *ActionLog* eine Aktion des Nutzers, wobei eine Instanz der Klasse *TimeLog* Informationen über eine vorgenommene Zeitmessung enthält. Für die Speicherung der Log-Einträge stellt die Bibliothek einen *ActionLogHolder* und einen *TimeLogHolder* zur Verfügung. Ein *ActionLog*-Eintrag kann einfach `ActionLogHolder.addActionLog()` hinzugefügt werden. Dabei werden Zeitstempel, Klassenname, Aktionsart, Umfrage ID und die ID der Frage gespeichert. Für die Zeitmessung wird die Klasse *TimeLogger* eingesetzt. Diese besitzt eine `startMeasurement()` und eine `stopMeasurement()` Funktion, die eine Zeitmessung starten und stoppen können. Wird eine Zeitmessung mit der Funktion `TimeLogger.startMeasurement()`

gestartet, wird eine Instanz der *TimeLog* Klasse erstellt. Dabei wird die Startzeit der Messung festgehalten. Wird die Zeitmessung anschließend über die Funktion `stopMeasurement()` gestoppt, wird die Endzeit ebenfalls in der erzeugten *TimeLog* Instanz gespeichert. Anschließend wird die erzeugte Instanz der Klasse *TimeLog* im *Singleton TimeLogHolder* gespeichert.

Für die Übermittlung der Log-Daten im *JSON*-Datenformat besitzen die Klassen *TimeLogHolder* und *ActionLogHolder* jeweils eine Funktion `toJSON()`, über die der Inhalt als *JSONObject* abgeholt werden kann. Für die spätere Rückkonvertierung bietet die Bibliothek zusätzlich eine Klasse *JSONLogParser* an. Diese bietet alle nötigen Funktionen zur Rückkonvertierung vom *JSON*-Datenformat in Objektinstanzen an. Die *LogLibrary* ist in Abbildung 5.4 als Klassendiagramm zu sehen.

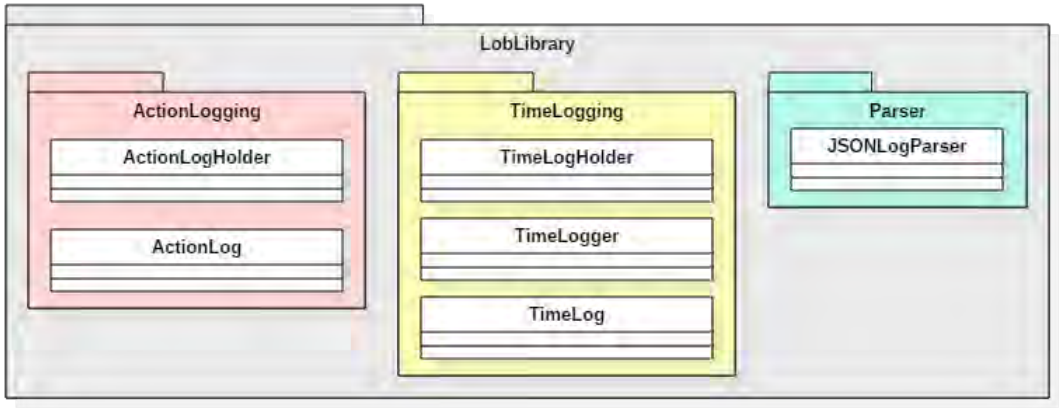

Abbildung 5.4: Klassendiagramm der *LogLibrary* Bibliothek

5.2 Webservice

Der entwickelte Webservice dient primär zur Bereitstellung von Umfragen und später zur Speicherung der für die Studie erfassten Daten. Der Webservice ist als *Java Dynamic Web Project* umgesetzt und implementiert die im Abschnitt 5.1 beschriebenen Bibliotheken *CrowdSensingLibrary* und *LogLibrary*. Umfragen werden vom Webser-

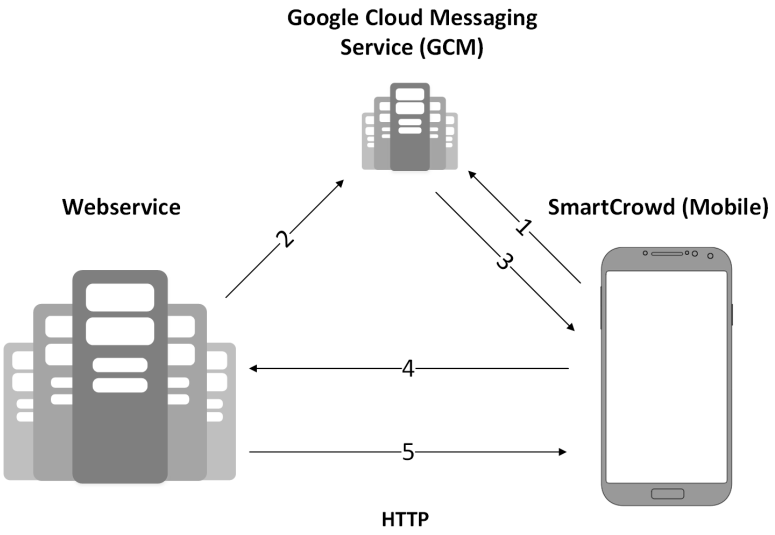

Abbildung 5.5: Kommunikationsablauf zwischen Webservice und *SmartCrowd (Mobile)*-Anwendung

vice im *JSON*-Datenformat bereitgestellt. Diese können über die `GET`-Methode von *HTTP* vom Webservice geladen werden. Später kann die ausgefüllte Umfrage über die `POST`-Methode wieder an den Webservice übergeben werden. Realisiert wird das über ein *Java Servlet* namens *QuestionnaireServlet*, das die Methoden `doGet()` und `doPost()` implementiert. Diese nehmen die Anfragen, an den Webservice, entgegen und versorgen den Client mit einer entsprechenden Antwortnachricht.

Möchte der Webservice seinen Benutzer mitteilen, dass neue Umfragen bereit stehen, so kann er das über den implementierten *Google Cloud Messaging Service* (GCM) tun. Dieser ermöglicht es *Push*-Benachrichtigungen an Geräte mit *Android* Betriebssystem zu senden. Realisiert wird das über die Einbindung des GCM in die im Kapitel 4 beschriebene Architektur. Die Abbildung 5.5 zeigt, wie das GCM integriert wird. Geräte, die über neue Umfragen informiert werden wollen, müssen sich beim GCM registrieren. Anschließend kann der Webservice über eine Nachricht dem GCM mitteilen, dass neue Umfragen verfügbar sind. Wird eine Umfrage an den Webservice geschickt, so wird die `doPost()` Methode des *QuestionnaireServlet* aufgerufen. Dort werden die im *JSON*-Datenformat übermittelten Ergebnisse des Fragebogens gespeichert.

Um die Auswertung der geplanten Studie zu erleichtern, soll der Webservice für jede an Ihn gesendete bearbeitete Umfrage die entsprechenden Ergebnisse und Log-Daten in einer Textdatei ablegen. Hierfür nutzt der Webservice die entwickelte *LogLibrary*. Die Log-Daten werden zusammen mit den bearbeiteten Fragebögen im *JSON*-Datenformat in der `doPost()` Methode entgegengenommen. Für die Speicherung der Log-Daten wurde die Klasse *LogResults* implementiert. Diese stellt die Schnittstelle der *LogLibrary* Bibliothek an den Webservice dar. Des Weiteren stellt der Webservice eine Klasse *LogFileWriter* zur Verfügung, die es erlaubt die entgegengenommenen Log-Daten in Textdateien auf dem Webservice zu speichern. Der Webservice speichert automatisch jede Umfrage mit dessen Ergebnissen, inklusive der Log-Daten, in einer Textdatei ab.

5.3 SmartCrowd (Mobile)

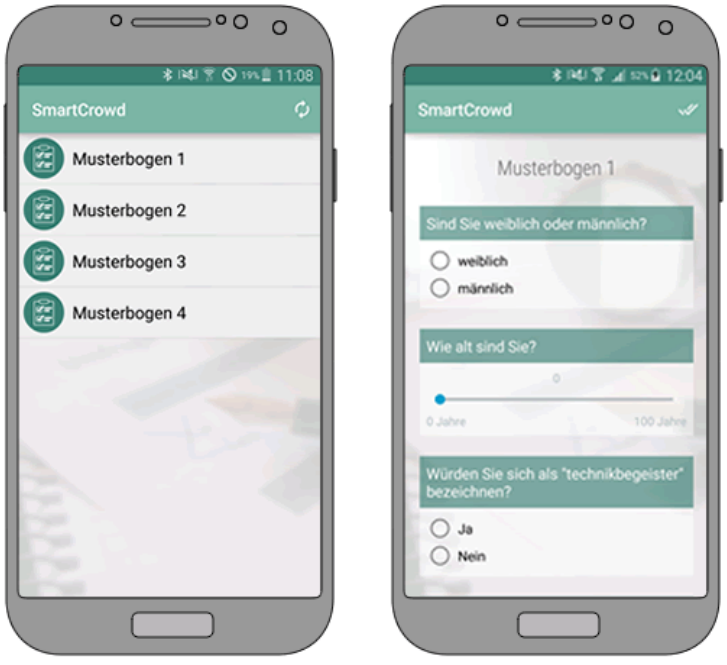

Abbildung 5.6: *SmartCrowd (Mobile)* a) *MainActivity* und b) *QuestionnaireActivity*

Die *SmartCrowd (Mobile)*- Anwendung für *Android Smartphones* fungiert als Anbindung der *SmartCrowd (Wear)*- Anwendung an den Webservice. Neben dieser Hintergrundlogik besitzt die *Smartphone*- Anwendung, wie Abbildung in 5.6 zu sehen, auch eine Benutzeroberfläche. Diese ist über die beiden *Activities MainActivity* und *QuestionnaireActivity* umgesetzt. Die *MainActivity* realisiert die Startseite der *SmartCrowd (Mobile)*-Anwendung und ist in Abbildung 5.6 a) zu sehen. Beim Start der Anwendung wird automatisch eine *HTTP*-Anfrage auf den Webservice ausgeführt, um die aktuellen Umfragen abzufragen. Alternativ kann der Benutzer auch über den in der Menüleiste befindlichen *Button* eine Anfrage an den Webservice starten. Die Benutzeroberfläche der *MainActivity* ist mit einer *ListView* realisiert. Sie stellt die Umfragen in Form von Listeneinträgen dar. Ein Listeneintrag enthält ein *Icon* und den Namen der Umfrage. Wählt der Benutzer eine Umfrage, durch klicken auf den entsprechenden Listeneintrag aus, wird die *QuestionnaireActivity* gestartet. Wie in Abbildung 5.6 b) zu sehen, repräsentiert diese die Darstellung einer Umfrage mit ihren Fragen und Antworten. Dabei werden die in der Bibliothek *CrowdSensingLibrary* definierten Fragearten unterstützt. Umgesetzt ist das in Abbildung 5.6 b) zu sehende Layout mit Hilfe eines *ScrollView*. Diese ist mit einzelnen *LinearLayouts* gefüllt, wobei jeweils ein *LinearLayout* eine Frage repräsentiert.

Die einzelnen Fragen können je nach Frageart über bereitgestellte Antwortoptionen, *Slider* oder Texteingabe über die *Smartphone*-Tastatur beantwortet werden. Sind alle Fragen beantworten, kann der Benutzer über den *Button* rechts in der Menüleiste die Ergebnisse an den Webservice senden.

Für die Entgegennahme von Push-Benachrichtigungen enthält die *SmartCrowd (Mobile)*-Anwendung die Klasse *PushGcmListenerService*, die einen *GcmListenerService* implementiert. Informiert der Webservice seine Benutzer über neue Umfragen über das GCM, wird die Methode `onMessageReceived(String form, final Bundle data)` in der Klasse *PushGcmListenerService* aufgerufen. Diese startet eine `GET`-Anfrage über *HTTP* an der Webservice und aktualisiert die Umfrageliste in der *MainActivity*. Die Abbildung 5.7 zeigt das Klassendiagramm der *SmartCrowd (Mobile)*- Anwendung.

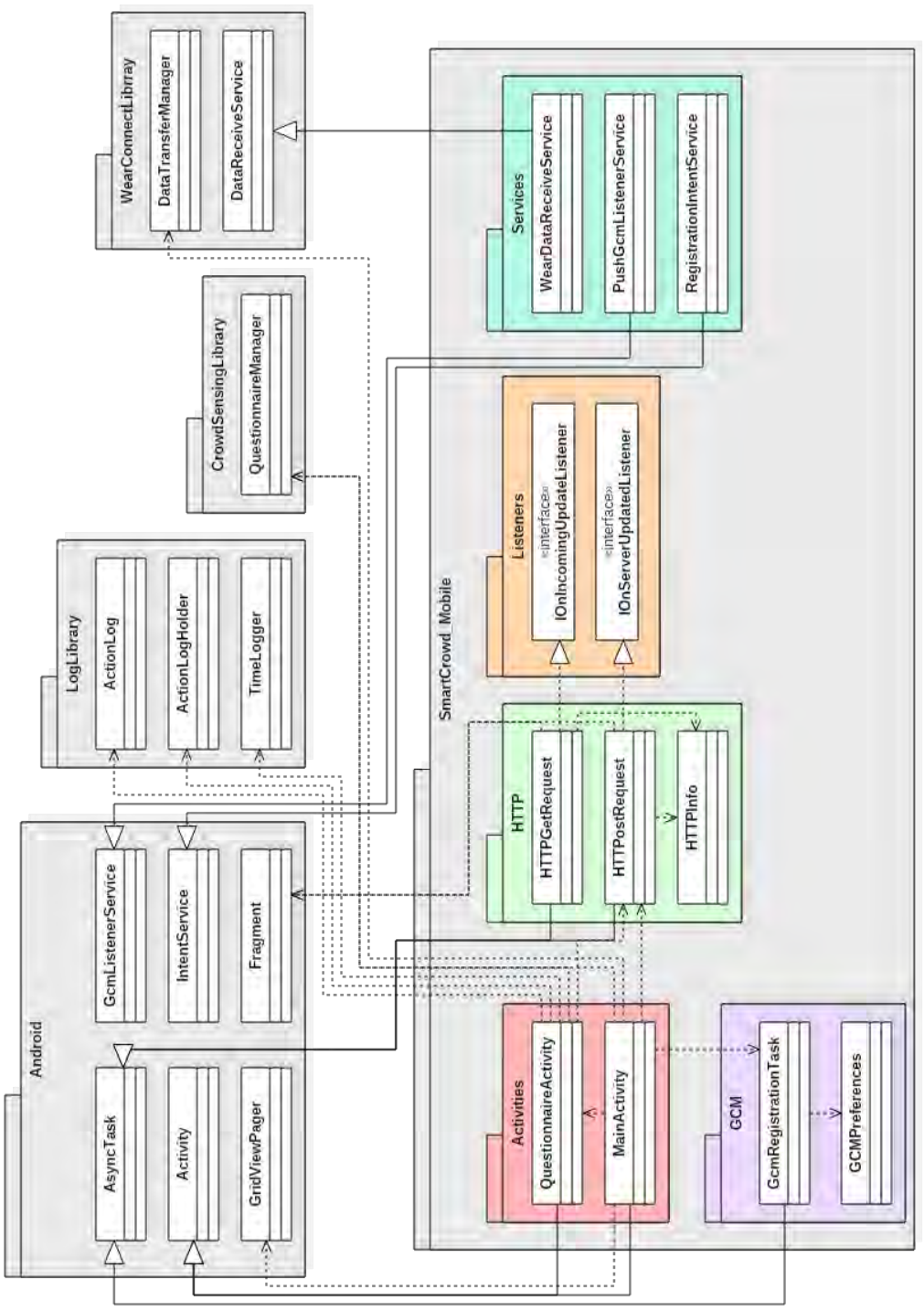

Abbildung 5.7: *SmartCrowd (Mobile)* Klassendiagramm

5.4 SmartCrowd (Wear)

Die Anwendung *SmartCrowd (Wear)* spielt in dieser Masterarbeit die zentrale Rolle, da mit dieser Anwendung die Tauglichkeit von *Android Wear* basierten *Smartwatches* für den Einsatz in *Mobile Crowd Sensing Systemen* geprüft werden soll. *SmartCrowd (Wear)* stellt eine *Android Wear* Anwendung auf Basis von *Android* 5.1 dar. Sie erlaubt es Umfragen im vollen Umfang zu bearbeiten und diese abschließend zur Auswertung an einen Webservice zu senden.

Zu Beginn wird im Abschnitt 5.4.1 das entwickelte Konzept zur Darstellung und Bearbeitung von Umfragen in einer *Android Wear*- Anwendung beschrieben. Im Weiteren wird auf die technische Implementierung der Anwendung eingegangen.

5.4.1 Umfrage-Designkonzept für Smartwatches

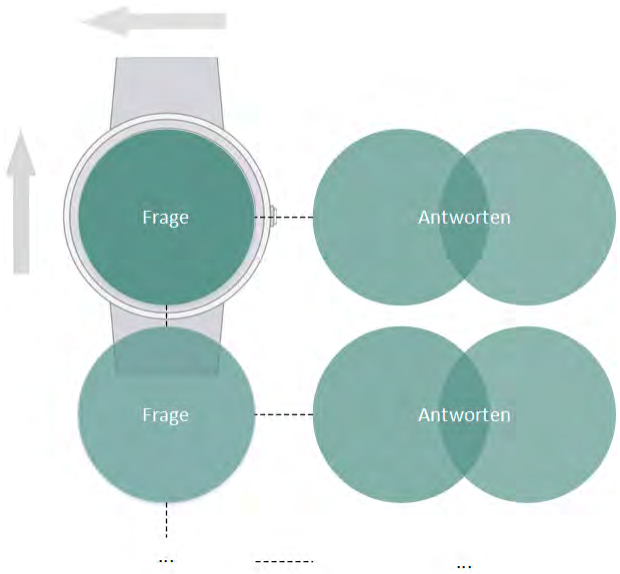

Abbildung 5.8: Umfrage-Designkonzept zur Darstellung und Bearbeitung von Umfragen auf einer *Android Wear* basierten *Smartwatch*

Smartwatches besitzen, bedingt durch ihre Größe ein sehr kleines Display für die Abbildung von Informationen. Für die Darstellung einer Frage inklusive deren Antwort-

möglichkeiten, fehlt auf der *Smartwatch* der Platz. Aus diesem Grund beinhaltet das in dieser Arbeit entwickelte Umfrage-Designkonzept, die Trennung von Frage und deren Antworten in gesonderte *Pages*.

Die Abbildung 5.8 zeigt beispielhaft, wie Umfragen mit dieser Anwendung auf der *Smartwatch* abgebildet werden. Der Benutzer bekommt zu Beginn einer Umfrage die erste Frage dargestellt. Mit einer Wischgeste nach links gelangt er zu den Antwort-*Pages*, die sich je nach Frageart unterschiedlich präsentieren. Umgesetzt wird das mit der Klasse *QuestionnaireGridViewPager*, die von dem in *Android Wear* zur Verfügung gestellten *GridViewPager* abgeleitet ist. Sie erlaubt es vertikal und horizontal über Wischgesten durch einzelne Fragen und dessen Antworten zu navigieren. Dabei stellt jede Reihe innerhalb des *QuestionnaireGridViewPager* eine Frage der Umfrage dar.

Dieses Prinzip der Interaktion mit einzelnen Fragen, entspricht auch den von Google empfohlenen Gestaltungsrichtlinien. Dort wird eine Wischgeste nach links, als Verlangen des Benutzers nach detaillierteren Informationen verstanden. Somit sind in dieser Anwendung detailliertere Informationen einer Frage als deren Antwort zu verstehen.

Im nächsten Abschnitt 5.4.2 wird die technische Umsetzung dieses Konzeptes in der *SmartCrowd (Wear)*- Anwendung vorgestellt.

5.4.2 Technische Implementierung

Die Abbildung 5.9 zeigt die Start-*Page*. Sie stellt dem Nutzer die verfügbaren Umfragen in Form einer Liste dar. Jeder Listeneintrag repräsentiert eine Umfrage, die der Benutzer über eine Berührung starten kann. Umgesetzt ist diese *Page* mit einer *WearableActivity* in Verbindung mit einer *WearableListView*. Diese Klassen sind speziell für *Smartwatches* optimierte Varianten der von *Android* bekannten *Activity* und der *ListView*.

Damit eine Anzeige der verfügbaren Umfragen möglich ist, implementiert die *WearableActivity* die *WearConnectLibrary* als auch die *CrowdSensingLibrary*. Über die *WearConnectLibrary* lädt sich die Anwendung beim Start, die zur Verfügung stehenden Umfragen aus dem Puffer der *Wearable.DataApi*. Hierbei kommt ein *ResultCallback* zum

Abbildung 5.9: Bild Startbildschirm *SmartCrowd (Wear)*

Einsatz, der alle *DataItems* im Puffer der *Wearable.DataApi* zurück liefert. Anschließend nutzt sie die *CrowdSensingLibrary* zur Verwaltung der Umfragen.

Startet der Benutzer eine Umfrage, wird eine neue *WearableActivity* namens *QuestionnaireActivity* gestartet. Diese verwaltet die komplette Umfrage auf Darstellungsebene. Für die Darstellung einer Umfrage und deren Fragen auf einer *Smartwatch*, wurde das im letzten Abschnitt 5.4.1 beschriebene Umfrage-Designkonzept in die *SmartCrowd (Wear)*- Anwendung integriert.

In den nun folgenden Abschnitten wird Schritt für Schritt erläutert, wie Fragen und Antworten innerhalb der *SmartCrowd (Wear)*- Anwendung dargestellt werden.

Darstellung von Fragen

Für die Darstellung einer Frage wurde die Klasse *QuestionFragment* implementiert. Sie leitet von der abstrakten Klasse *CardFragment* ab und beinhaltet im Wesentlichen eine *Card*. Diese *Card* wird in dieser Anwendung für die Darstellung von Fragen herangezogen. Zusätzlich besitzt die *Card* an der rechten oberen Kante einen *Button*. Durch drücken dieses *Buttons* ist es dem Benutzer möglich, die Umfrage an die *Smartphone* Anwendung *SmartCrowd (Mobile)* zu delegieren. Damit ist es dem Benutzer nach jeder Frage möglich die Bearbeitung des Fragebogens auf dem *Smartphone* fortzusetzen. Die

Abbildung 5.10 zeigt die Darstellung einer Frage über die Klasse *QuestionFragment*. Das *QuestionFragment* stellt für jede Frage im *QuestionnaireGridViewPager* die einleitende *Page* dar.

Abbildung 5.10: Darstellung einer Frage auf der *Smartwatch*

Infolge der Verwendung des *QuestionFragment* für jede Frage benötigt die Anwendung weitere *Fragments* für die Präsentation und Eingabe von Antworten. Die Anwendung soll die in der Tabelle 3.1 definierten Fragearten unterstützen. Hierfür wurden entsprechende Eingabeformen entwickelt. Die Tabelle 5.1 benennt die entwickelten Eingabeformen und ordnet diese den Fragearten zu. Die entwickelte Anwendung bietet für die Fragearten *Multiple Choice*, *Single Choice* und Skala jeweils zwei verschiedene Eingabeformen an. Diese sollen später in der geplanten Studie gegeneinander evaluiert werden. In den kommenden Abschnitten werden die in der Tabelle 5.1 benannten Eingabeformen beschrieben.

Frageart	**Eingabeformen**
Multiple Choice	*MultipleChoiceAnswerList*, *MultipleChoiceAnswerCard*
Single Choice	*SingleChoiceAnswerList*, *SingleChoiceAnswerCard*
Skala	*LockScale*, *ButtonScale*
Offene Fragen	*TextAnswerCard*

Tabelle 5.1: Zuordnung von Fragearten zu Eingabeformen

Darstellung von Antworten - MultipleChoiceAnswerList

Die Eingabeform *MultipleChoiceAnswerList* wurde zur Beantwortung von *Multiple Choice* Fragen entwickelt. Das gleichnamige *Fragment* leitete von der Klasse *Fragment* ab und realisiert die Benutzeroberfläche dieser Eingabeform. Das dazugehörige Layout implementiert eine *List* mit Hilfe der Klasse *WearableListView*. Für die Darstellung einer einzelnen Antwortoption wurde die Klasse *MultipleChoiceItemViewHolder* implementiert. Jede Antwortoption wird in der *List* zusammen mit einer *Checkbox* dargestellt. Zur Erkennung einer Auswahl durch den Benutzer wird ein *OnCheckedChangedListener* implementiert, der den *MultipleChoiceItemViewHolder* über die Zustandsänderung informiert. Der *QuestionnaireManager* der *CrowdSensingLibrary* ist dem *MultipleChoiceItemViewHolder* bekannt und wird von ihm über Änderungen informiert. Die Abbildung 5.11 zeigt den Aufbau einer *Multiple Choice* Frage unter Verwendung des *MultipleChoiceAnswerList Fragment* .

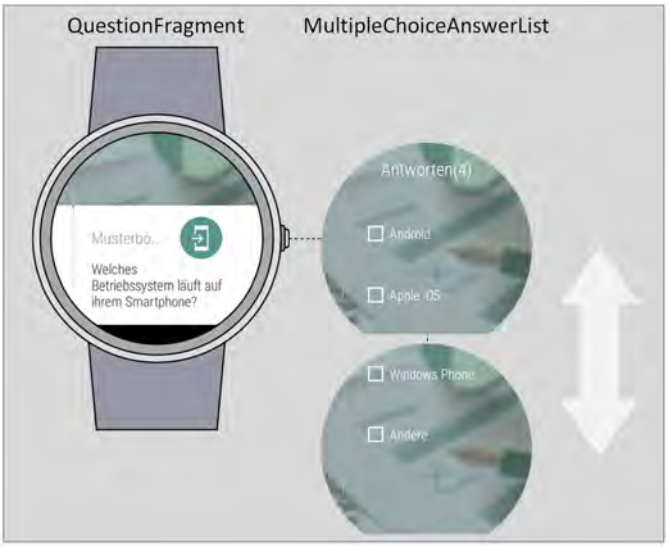

Abbildung 5.11: *Multiple Choice* Frage unter Verwendung des *MultipleChoiceAnswerList Fragment*

Darstellung von Antworten - MultipleChoiceAnswerCard

Auch die Eingabeform *MultipleChoiceAnswerCard* ist zur Beantwortung von *Multiple Choice* Fragen entwickelt worden. Sie bietet eine Alternative zur *MultipleChoiceAnswerList* Eingabeform. Bei dieser Eingabeform, werden die einzelnen Antwortoptionen jeweils als eigene *Card* dargestellt. Für jede Antwortoption wird ein *MultipleChoiceAnswerCard Fragment* erstellt. Jede *Card* beinhaltet eine *Checkbox* mit der dazugehörigen Antwortoption. Die resultierenden *Fragments* werden anschließend in den *QuestionnaireGridViewPager* integriert. Die Abbildung 5.12 zeigt eine *Multiple Choice* Frage unter Verwendung von *MultipleChoiceAnswerCard Fragments*.

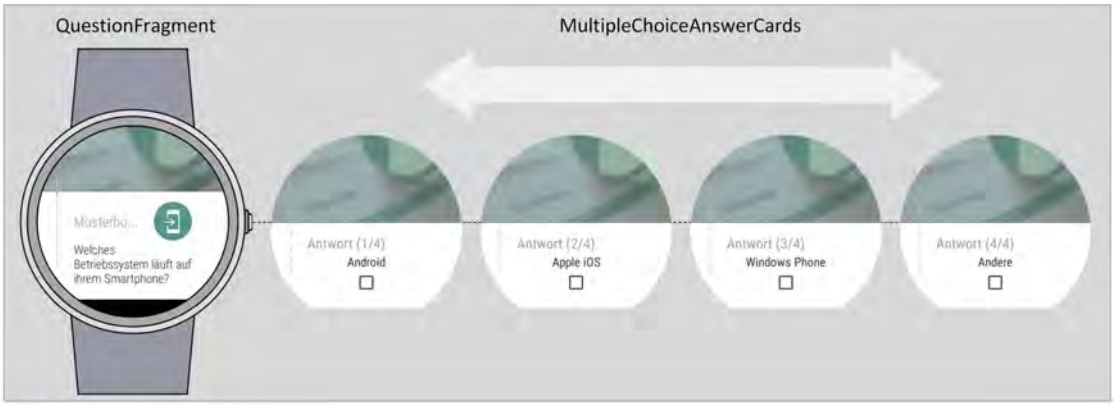

Abbildung 5.12: *Multiple Choice* Frage unter Verwendung von *MultipleChoiceAnswerCard Fragments*

Darstellung von Antworten - SingleChoiceAnswerList

Dieses *Fragment* implementiert die Eingabeform für Antworten einer *Single Choice* Frage. Zur Darstellung der Antwortoptionen wird ein *SingleChoiceItemViewHolder* implementiert, der jeder Antwortoption eine *Radiobox* zuordnet. Zur Erkennung einer Benutzerauswahl wird ein *OnCheckedChangedListener* eingesetzt, der den *SingleChoiceItemViewHolder* über die Zustandsänderung informiert. Auch hier wird die Änderung unmittelbar vom *QuestionnaireManager* übernommen. Die Abbildung 5.13 zeigt eine *Single Choice* Frage unter Verwendung des *SingleChoiceAnswerList Fragment*.

Abbildung 5.13: *Single Choice* Frage unter Verwendung des *SingleChoiceAnswerList Fragment*

Darstellung von Antworten - SingleChoiceAnswerCard

Dieses *Fragment* implementiert eine alternative Eingabeform für Antwortoptionen von *Single Choice* Fragen für die Eingabeform *SingleChoiceAnswerList*. Ähnlich wie beim *MultipleChoiceAnswerCard Fragment*, werden die Antwortoptionen jeweils als eigene *Card* dargestellt. Jedoch ist bei dieser Variante die Auswahl nur einer Antwortoption zugelassen. Um flüssige Übergänge zwischen einzelnen *Pages* zu gewährleisten, puffert das *Android Wear* Betriebssystem benachbarte *Pages*. Das kann dazu führen, dass bei der wechselnden Selektierung von Antwortoptionen, Änderungen an der Benutzeroberfläche der einzelnen *Cards* nicht übernommen werden. Zur Umgehung dieses Problems wurde ein *OnCheckedChangedListener* implementiert. Durch diesen *Listener* werden alle *SingleChoiceAnswerCard Fragments* miteinander bekannt gemacht um sich bei Änderungen gegenseitig zu informieren. Damit können beim Setzen einer *Radiobox* alle anderen *Fragments* dazu aufgefordert werden, ihre *Radiobox* zurückzusetzen. Die Abbildung 5.14 zeigt eine *Single Choice* Frage unter Verwendung von *SingleChoiceAnswerCard Fragments*.

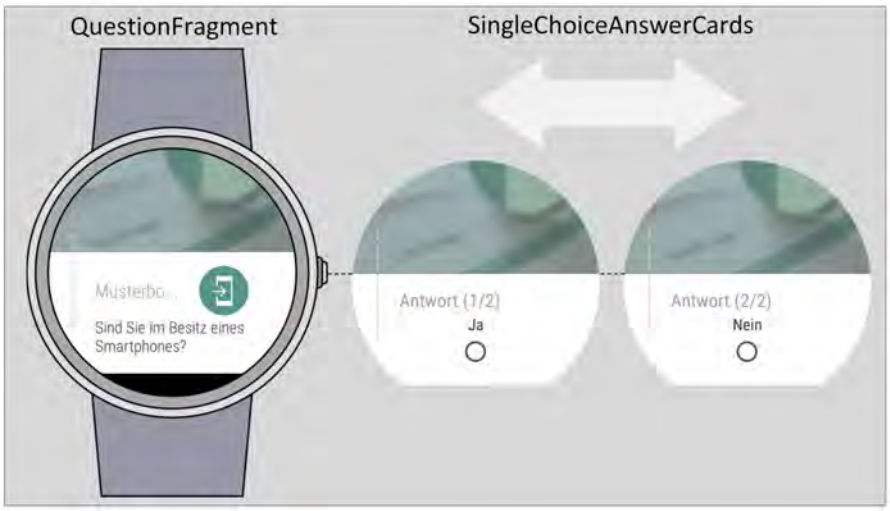

Abbildung 5.14: *Single Choice* Frage unter Verwendung der Eingabeform *SingleChoiceAnswerCard Fragments*

Darstellung von Antworten - LockScale

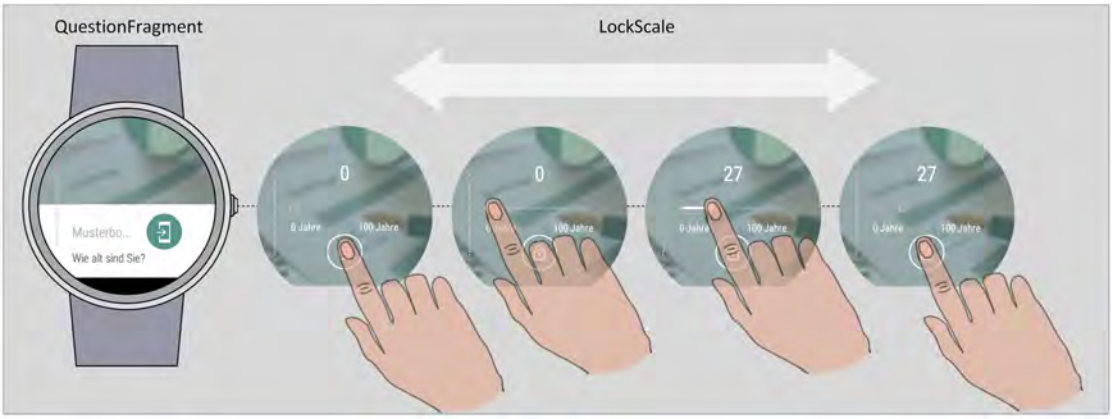

Abbildung 5.15: Eingabeform *LockScale* zur Beantwortung einer Frage über eine Skala

Das *LockScale Fragment* bietet die Möglichkeit zur Eingabe von Zahlenwerten. Wie in Abbildung 5.15 zu sehen ist stellt diese Eingabeform eine *Seekbar* zur Auswahl eines Wertes auf einer Skala bereit. Die *Seekbar* benötigt zur Auswahl eines Wertes auf der Scala die Wischgesten nach links und rechts. Diese Gesten werden allerdings vom

QuestionnaireGridViewPager zur Interaktion zwischen verschiedenen *Pages* verwendet, was den Einsatz einer *Seekbar* innerhalb eines *GridViewPagers* erschwert. Um dieses Problem zu lösen wird der sogenannte *Lock-Button* eingeführt. Dieser erlaubt es, die Gestenerkennung des *QuestionnaireGridViewPager* auszusetzen und die *Seekbar* zu aktivieren. An dieser Stelle kann der Benutzer einen Wert über die Seekbar auswählen. Er kann jedoch nicht die Page mit einer Wischgeste verlassen. Erst durch ein weiteres Drücken des *Lock-Button* wird die *Seekbar* wieder deaktiviert und die Gestenerkennung des *QuestionnaireGridViewPager* aktiviert. Danach ist das Verlassen der *Page* über eine Wischgeste wieder möglich ist. Google schreibt in seinen Richtlinien vor, dass es einem Benutzer zu jeder Zeit möglich sein muss, eine Anwendung zu verlassen. Aus diesem Grund wurde eine *Wearable.View.DismissOverlayView* implementiert. Diese wird geöffnet, wenn der Benutzer den Bildschirm längere Zeit gedrückt hält. Innerhalb dieser View wird dem Benutzer ein *Button* zur Verfügung gestellt, über den die Anwendung geschlossen werden kann.

Darstellung von Antworten - ButtonScale

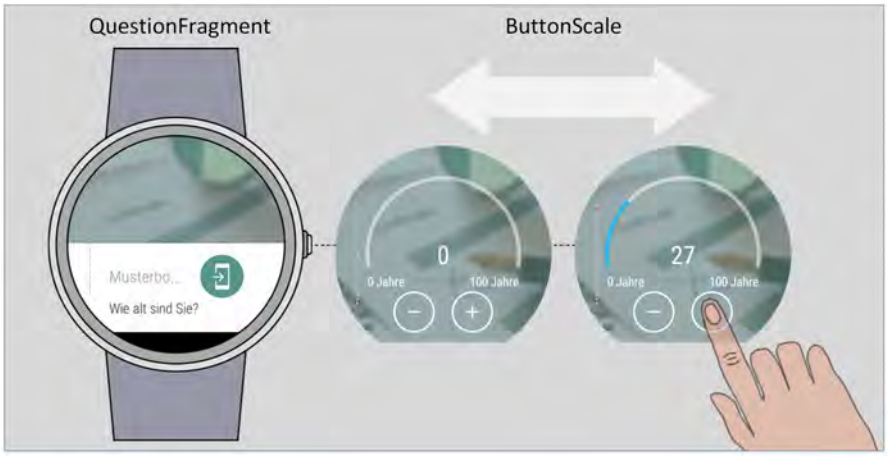

Abbildung 5.16: Eingabeform *ButtonScale* zur Beantwortung einer Frage über eine Skala

Alternativ zur Eingabeform *LockScale* bietet die Anwendung die Möglichkeit Skalen über die Eingabeform *ButtonScale* zu beantworten. Dieser Ansatz verzichtet auf die Interaktion mit Gesten und bietet als Alternative zwei *Buttons* an. Mit diesen kann der

Benutzer auf der Skala navigieren und einen Wert selektieren. Dabei ist der rechte *Button* für die positive Richtung und der linke *Button* für die negative Richtung zu verwenden. Mit den *Buttons* kann über die Grenzen hinweg navigiert werden. Das bedeutet, wenn durch Halten des +-*Button* der maximale Wert erreicht ist, springt die Skala wieder zum Anfangswert zurück. Außerdem wurde eine sogenannte *Spull*-Funktion implementiert. Diese erlaubt es bei längerem Drücken ein schnelleres Fortschreiten auf der Skala. Das kann bei einem großen Wertebereich zu einer Zeitersparnis führen. Die Abbildung 5.16 zeigt eine Frage, die über die Eingabeform *ButtonScale* beantwortet werden kann.

Darstellung von Antworten - TextAnswerCard

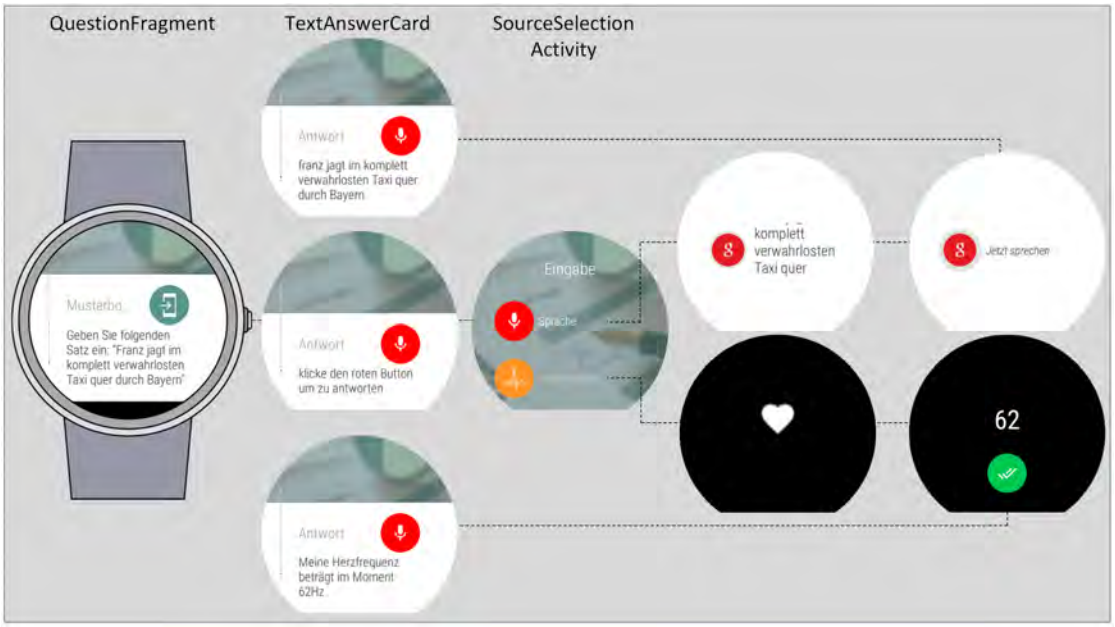

Abbildung 5.17: Beantwortung von offenen Fragen mittels der Eingabeform *TextAnswerCard*

Für die Beantwortung von offenen Fragen wurde die Eingabeform *TextAnswerCard* implementiert. Die Texteingabe funktioniert unter *Android Wear* ausschließlich über die von Google integrierte Spracheingabe, die auch in dieser Anwendung zum Einsatz kommt. Das *TextAnswerCard Fragment* leitet von der abstrakten Klasse *CardFrag-*

ment ab und beinhaltet eine *Card*. Diese *Card* stellt beim ersten Start Hinweise zur Spracheingabe bereit. Mit dem roten *Button* in der rechten oberen Ecke kann die Spracheingabe gestartet werden. Dabei wird der von Google zur Verfügung gestellte *RecognizerIntent.ACTION_RECOGNIZER_SPEECH* verwendet. Für Geräte mit einem Herzfrequenzsensor wird beim Betätigen des *Button* eine Auswahlliste angezeigt. Dort kann der Benutzer auch seine aktuelle Herzfrequenz als Antwort verwenden. Dies bietet beispielsweise bei medizinischen Umfragen neue Möglichkeiten der Datenerfassung. Bei Geräten die keinen solchen Sensor besitzen, wird sofort die Spracheingabe gestartet. In Abbildung 5.17 ist der Ablauf für die Beantwortung einer offenen Frage mit den implementierten Funktionen dargestellt.

Abschließen und Speichern einer Umfrage

In den vorigen Abschnitten wurde die Eingabeformen beschrieben, die von der *Smart-Crowd (Wear)*- Anwendung unterstützt werden. Am Ende einer Umfrage wird auf der *Smartwatch* ein abschließendes Element benötigt. Die Klasse *FinishFragment* bildet dieses abschließende Element. Dieses *Fragment* dient dazu, die Umfrageergebnisse an den Webservice zu übermitteln und die Umfrage zu beenden. Das *FinishFragment* leitet von der abstrakten Klasse *Fragment* ab und beinhaltet, wie in Abbildung 5.18 zu sehen, einen einzelnen *Button*. Durch betätigen des *Button* wird die Umfrage abgeschlossen und die Ergebnisse an den Webservice übermittelt. Der Benutzer wird auf die Start-*Page* der Anwendung geleitet.

Abbildung 5.18: Speichern-*Button* am Ende einer Umfrage

5.4.3 Benachrichtigungen

Abbildung 5.19: Benachrichtigungen über neue Fragebögen auf einer *Smartwatch*

Über die aktive Verbindung zum *Smartphone* und der *SmartCrowd (Mobile)*- Anwendung, können die über die *CrowdSensingLibrary* empfangenen Benachrichtigungen auf der *Smartwatch* angezeigt werden. Die Abbildung 5.19 zeigt eine Benachrichtigung auf der *Smartwatch*. Benachrichtigungen auf der *Smartwatch* enthalten grundsätzlich die Aktionen “Auf Telefon öffnen“, “Öffnen“ und “Blockieren“. Dabei kann über die Aktion “Öffnen“ die *SmartCrowd (Wear)*- Anwendung auf der *Smartwatch* geöffnet werden. Die Aktion “Auf Telefon öffnen“ hingegen öffnet die *SmartCrowd (Mobile)*- Anwendung auf dem *Smartphone*. Mit der Aktion “Blockieren“ können die Benachrichtigungen auf der *Smartwatch* für die Anwendung *SmartCrowd (Wear)* abgeschaltet werden.

Informiert der Webservice seine Benutzer über neue Umfragen durch das GCM, so wird diese Benachrichtigung auch auf der *Smartwatch* angezeigt. Realisiert ist das, durch eine Erweiterung der Klasse *PushGcmListenerService* in der *SmartCrowd (Mobile)*- Anwendung. Diese wird um einen Aufruf des *DataTransferManager* der *WearConnectLibrary* erweitert. Die Erweiterung bewirkt, dass die Benachrichtigung über die *Wearable.DataApi* an die *SmartCrowd (Wear)*- Anwendung auf der Uhr weitergeleitet wird. Dort wird sie von der *WearConnectLibrary* entgegengenommen. Abschließend wird eine lokale Benachrichtigung erzeugt und auf der Uhr angezeigt. Entscheidet sich der Benutzer für die Aktion “Auf *Smartphone* Öffnen“ so wird über die *Wearable.DataApi* die *Smart-*

Crowd (Mobile)- Anwendung auf dem *Smartphone* der Befehl zum Öffnen gegeben. Die Abbildung 5.20 komplettiert das in Kapitel 4 vorgestellte Benachrichtigungskonzept für die Unterstützung von *Android Wear- Smartwatches*.

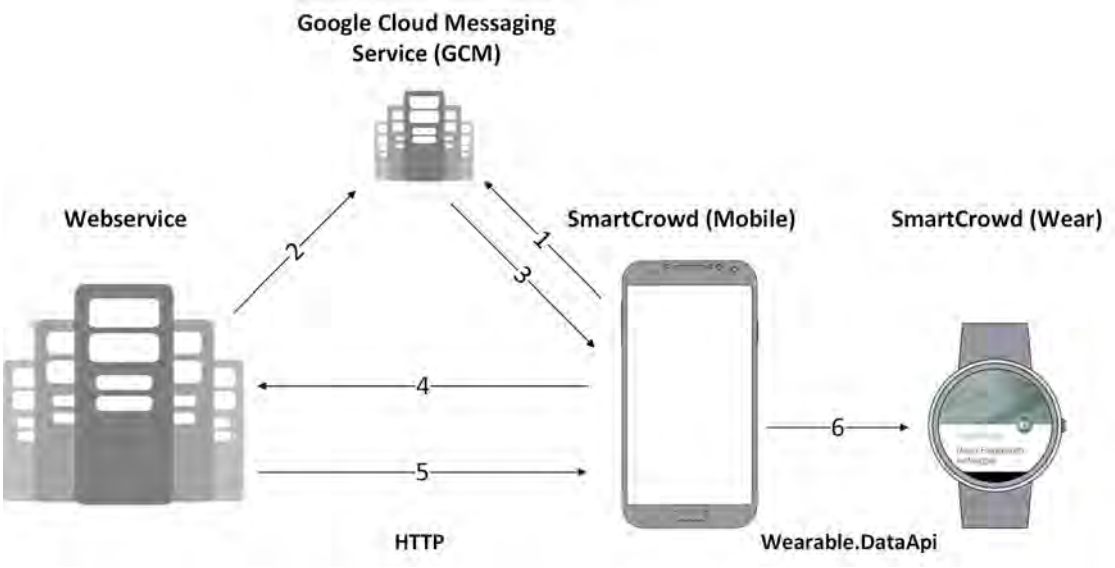

Abbildung 5.20: Benachrichtigungen - Nachrichtenverlauf

5.4.4 Fortschrittsanzeige

Die *Android* Anwendung *SmartCrowd (Mobile)* bietet dem Benutzer anhand des *Scrollbar* der *ScrollView* eine Art Übersicht über den Fortschrittsprozess der Umfrage an. Somit kann der Benutzer erkennen, an welcher Stelle er sich gerade innerhalb der Umfrage befindet. Unter *Android Wear* gibt es hierfür kein Standardkonzept, weshalb für diese Anwendung ein eigenes Konzept zur Darstellung des Fortschrittprozesses implementiert wurde. Jede Reihe innerhalb des *QuestionnaireGridViewPager* wird dabei als weißer Punkt am linken Bildschirmrand dargestellt. Der Punkt, welcher die aktuell sichtbare Reihe darstellt, wird rot markiert. Hiermit bietet auch die *SmartCrowd (Wear)*-Anwendung die Möglichkeit dem Nutzer einen Überblick über den Fragebogen zu geben. Umgesetzt wurde dieses Konzept über ein transparentes *FrameLayout*, das auf den anderen *Layouts* aufgelegt ist.

5.5 Implementierungserweiterung für die Studie

Zur Darstellung der geplanten Studie ist es erforderlich, dass die beiden Anwendungen *SmartCrowd (Mobile)* und *SmartCrowd (Wear)* um die Implementierung der *LogLibrary* aus Abschnitt 5.1.3 erweitert werden.

Die *SmartCrowd (Mobile)*- Anwendung wird dahingehend erweitert, dass eine Zeitmessung zur Ermittlung der Gesamtbearbeitungszeit einzelner Umfragen durchgeführt werden kann. Zudem wird die Zeit gemessen, wie lange der Benutzer zur Beantwortung der offenen Fragen mit der *Smartphone* Tastatur benötigt hat.

In der *SmartCrowd (Wear)*- Anwendung wird ebenfalls die Gesamtbearbeitungszeit eines Fragebogens gemessen. Hierzu wird die Zeit zwischen dem Eintritt in die Umfrage und dem Drücken des Speichern *Buttons* herangezogen. Zusätzlich wird die Bearbeitungszeit jeder einzelnen Frage dokumentiert. Dafür wird die Zeit zwischen Eintreten und Austreten in eine Reihe des GridViewPagers ermittelt und der Frage zugeordnet. Des Weiteren werden in der *SmartCrowd (Wear)*- Anwendung die Interaktionen des Benutzers auf dem Bildschirm aufgezeichnet.

Die dadurch erzeugten Log-Einträge werden am Ende einer Umfrage zusammen mit den Umfrageergebnissen an den Webservice übertragen. Dort werden sie als Textdatei gespeichert.

Abschließend zeigt die Abbildung 5.21 eine Übersicht der *SmartCrowd (Wear)*- Anwendung als Klassendiagramm. Das Paket *SmartCrowd_ Wear* bildet dabei die *SmartCrowd (Wear)*- Anwendung ab.

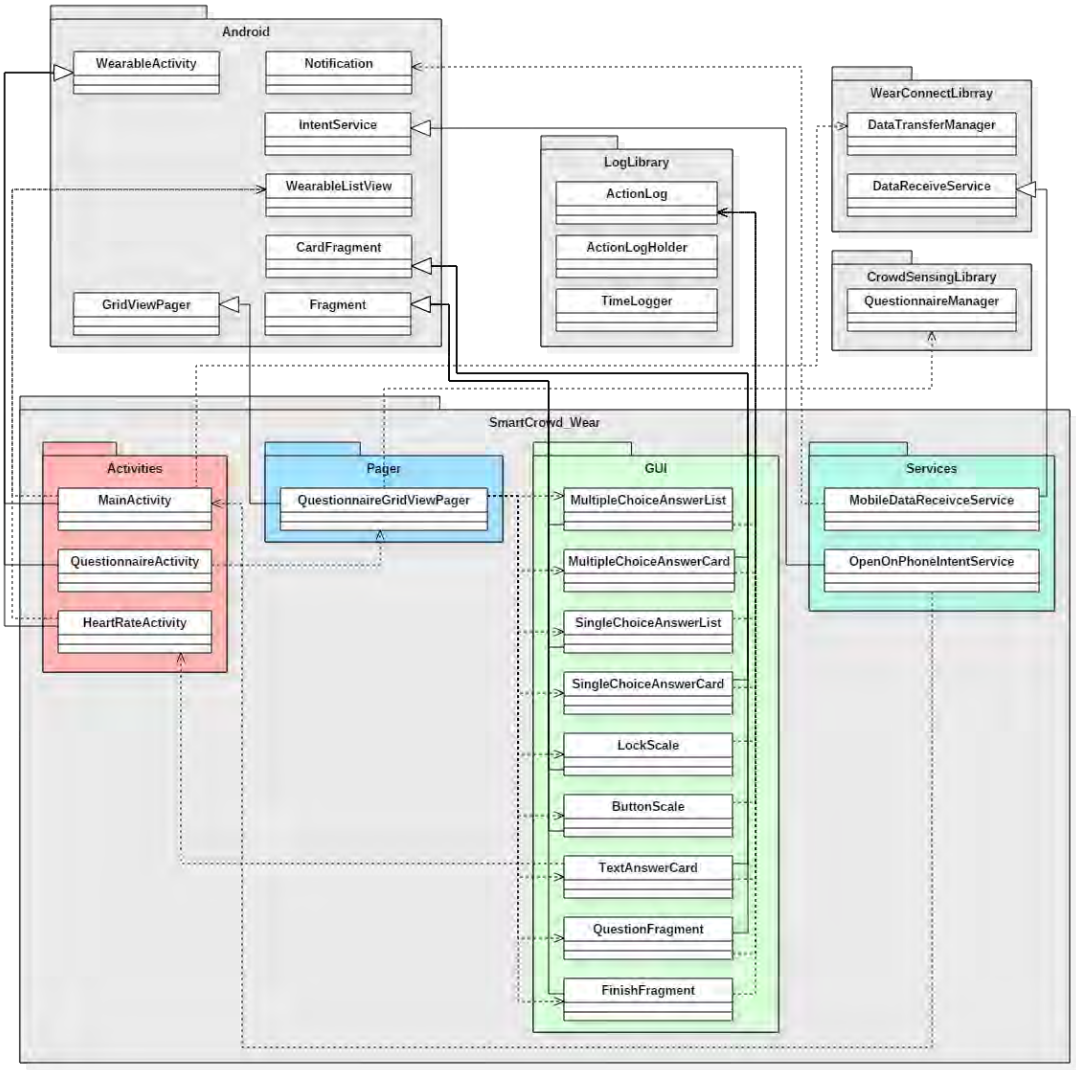

Abbildung 5.21: Klassendiagramm der *SmartCrowd (Wear)*- Anwendung

6

Studie

In diesem Kapitel wird die durchgeführte Studie vorgestellt. Ziel der Studie ist es, die implementierten Eingabeformen zu evaluieren und die *SmartCrowd (Wear)*- Anwendung selbst auf die Tauglichkeit für den Einsatz in einem *Mobile Crowd Sensing System* zu überprüfen.

In der nachfolgenden Tabelle 6.1 werden die aufgestellten Hypothesen vorgestellt. Im Abschnitt 6.1 wird der Aufbau der Studie erläutern. Es folgen die Beschreibung der Durchführung und die Ergebnisse der Studie.

Nr.	Hypothese
1.	*Smartwatches* können mittels einer optimierten Anwendung in *Mobile Crowd Sensing Systemen* eingesetzt werden.
2.	Das Ausfüllen von Umfragen mit vielen Fragen auf einer *Smartwatch* ist für den Benutzer anstrengend.
3.	Der Benutzer bevorzugt zum Ausfüllen von Umfragen das *Smartphone*.
4.	Damit ein Benutzer eine Umfrage vollständig auf der *Smartwatch* bearbeitet, darf die Umfrage nicht mehr als 4-6 Fragen enthalten.
5.	Die Mehrheit der Benutzer würde Fragen bezüglich ihrer Gesundheit in öffentlichen Verkehrsmitteln nicht per Spracheingabe beantworten.
6.	Fragen zur Gesundheit des Benutzers, werden auf der *Smartwatch* als auch auf dem *Smartphone* mit einem weniger guten Gefühl beantwortet.

Fortsetzung auf Folgeseite

Tabelle 6.1: Hypothesen

Nr.	Hypothese
7.	Die implementierte Fortschrittsanzeige erachtet der Benutzer als hilfreich, um seine aktuelle Position innerhalb des Fragebogens zu kennen.
8.	Durch Integration der Spracheingabe können offene Fragen bequem über die *Smartwatch* beantwortet werden.
9.	Zur Beantwortung von *Multiple Choice* und *Single Choice* Fragen wird die Eingabeform *Cards* bevorzugt.
10.	Die *LockScale* ist keine akzeptable Eingabeform für exakte Werte.
11.	Die Bearbeitungszeit einer Frage steigt mit zunehmender Anzahl an Antwortoptionen bzw. Stufen.

Tabelle 6.1: Hypothesen

6.1 Aufbau

Nachfolgend wird der Aufbau der Studie beschrieben. Im ersten Abschnitt wird mit der Struktur der Studie begonnen. Anschließend werden die beiden Fragebogenarten Musterfragebogen und Vergleichsfragebogen vorgestellt.

6.1.1 Struktur

Zu Beginn der Studie werden die Probanden anhand einer einleitenden Präsentation an die Studie herangeführt. Die Präsentation besteht aus drei Teilen. Da die meisten Probanden noch nie mit einer *Smartwatch* interagiert haben, werden diese im ersten Teil der Präsentation über die wesentlichen Interaktionskonzepte von *Android Wear* aufgeklärt. Anschließend wird ihnen im zweiten Teil, die *SmartCrowd (Wear)*- Anwendung mit ihren Eingabeformen und die *SmartCrowd (Mobile)*- Anwendung vorgestellt. Im dritten Teil wird Ihnen gesagt, dass Sie die Uhr am Handgelenk ihrer Wahl anlegen und diese bis zum Ende der Studie dort belassen sollen. Zudem werden Sie darüber aufgeklärt, dass sie während der Studie mit einer Videokamera aufgezeichnet werden. Die Präsentation befindet sich im Anhang A und kann dort eingesehen werden.

Die Studie baut auf das von Scott MacKenzie im Jahr 2009 vorgestellte *Within Subject Design* [27]. Dabei bearbeitet jeder Proband den Musterfragebogen auf der *Smartwatch* unter Verwendung der unterschiedlichen Eingabeformen. Um die Bearbeitungsdauer zwischen *Smartphone* und *Smartwatch* vergleichen zu können bearbeitet jeder Proband den Musterfragebogen ein weiteres Mal auf dem *Smartphone*. Um Lerneffekte auszugleichen, kommt das sogenannte *Counterbalancing* Verfahren zur Anwendung. Das bedeutet, dass die Probanden die einzelnen Eingabeformen *MultipleChoiceAnswerList*, *SingleChoiceAnswerList*, *MultipleChoiceAnswerCard*, *Single-ChoiceAnswerCard*, *LockScale* und *ButtonScale* in unterschiedlicher Reihenfolge durchlaufen und die Geräte in unterschiedlicher Reihenfolge verwenden. Hierzu werden 24 Probanden per Losverfahren in 6 Gruppen mit jeweils vier Probanden eingeteilt. Die Gruppeneinteilung entscheidet, ob der Proband mit dem *Smartphone* oder mit der *Smartwatch* beginnt. Darüber hinaus bestimmt das Los, in welcher Reihenfolge die Eingabeformen auf der *Smartwatch* durchlaufen werden.

Während der Studie werden die Antworten und alle erforderlichen Zeitmessungen sowie Interkationen innerhalb der *SmartCrowd (Wear)*- Anwendung aufgezeichnet und protokolliert. Die *SmartCrowd (Mobile)*- Anwendung zeichnet zusätzlich zu den Antworten der Probanden auch die Bearbeitungsdauer für einen ganzen Fragebogen auf. Dies dient später dazu, die Bearbeitungsdauer eines Fragebogens auf dem *Smartphone* und der *Smartwatch* zu vergleichen. Für die Protokollierung wird die in Kapitel 5.1.3 vorgestellte *LogLibrary* verwendet.

Für den weiteren Verlauf der Studie werden die Eingabeformen auf der *Smartwatch* in vier Eingabeformversionen gruppiert. Dabei werden die Eingabeformen *MultipleChoiceAnswerList* und *SingleChoiceAnswerList* sowie *MultipleChoiceAnswerCard* und *Single-ChoiceAnswerCard*, aufgrund ihrer Ähnlichkeit, gemeinsam evaluiert. Im weiteren Verlauf dieser Arbeit werden diese Eingabeformen verkürtzt als *List* und *Cards* bezeichnet. Eine weitere Version stellt die Bearbeitung des Musterfragebogens auf dem *Smartphone* dar. Die Abbildung 6.1 zeigt die Gruppierung der Eingabeformen in fünf Versionen. Dabei werden die unterschiedlichen Anordnungen der Eingabeformen als V1 bis V4 bezeichnet. Die Version mit dem *Smartphone* erhält den Namen V5.

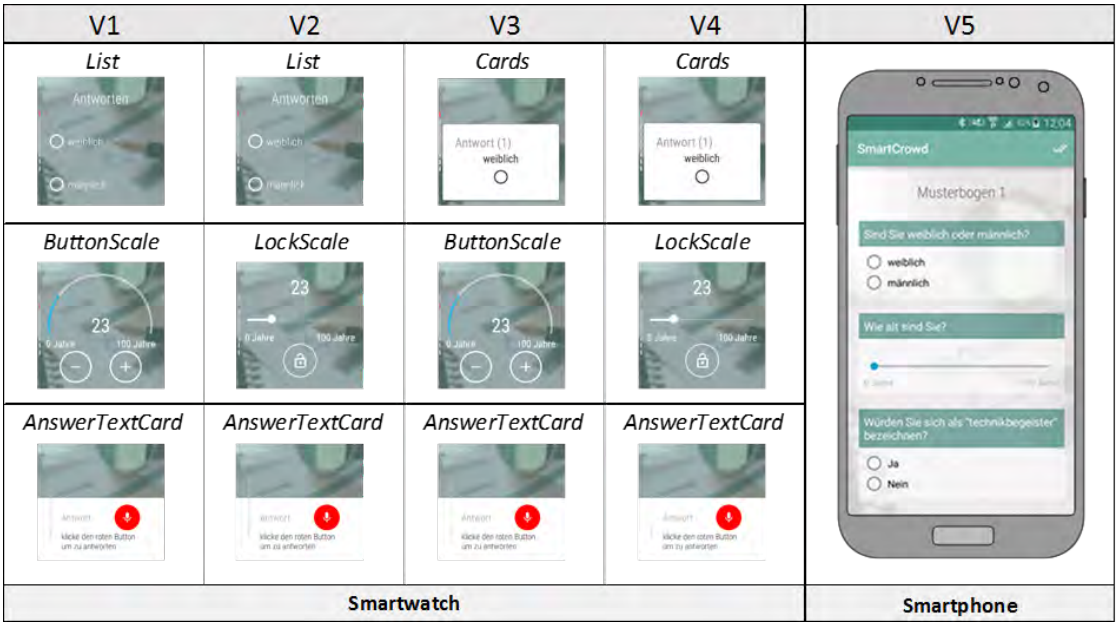

Abbildung 6.1: Gruppierung der Eingabeformen in fünf Versionen

Hat der Proband alle Versionen durchlaufen, wird ihm abschließend der Vergleichsfragebogen zum Ausfüllen vorgelegt.

6.1.2 Musterfragebogen

Der Musterfragebogen ist der Fragebogen, den die Probanden auf den einzelnen Geräten ausfüllen müssen. Er ist in insgesamt sechs Blöcke mit unterschiedlichen Fragearten unterteilt. Im ersten Block befinden sich Fragen zu den Eigenschaften der Person, wie beispielsweise Alter, Geschlecht und Nutzungsverhalten von *Smartphones* und *Smartwatches*. In den nachfolgenden Blöcken zwei bis fünf befinden sich Fragen aus einem deutschen WHO Fragebogen zum allgemeinen Gesundheitszustand einer Person. Diese Fragen werden dazu verwendet um die entwickelten Eingabeformen zu testen und zu evaluieren. Dazu werden in Block zwei und vier jeweils drei *Single Choice* Fragen gestellt, die mit drei, fünf und sieben Antwortmöglichkeiten versehen sind. Die Blöcke drei und fünf enthalten ebenfalls drei Fragen, die allerdings über eine Skala beantwortet werden sollen. Auch hier werden drei, fünf und sieben Abstufungen verwendet, die auf

der Skala selektiert werden können. Im sechsten und letzten Block befinden sich drei offene Fragen. Mit diesen Fragen soll ein Vergleich zwischen der Spracheingabe auf der *Smartwatch* und der Eingabe über die *Smartphone*-Tastatur ermöglicht werden. Um die ermittelten Zeiten vergleichbar zu machen, werden die Probanden in diesen Fragen dazu aufgefordert einen vorgegebenen Text mit der vorgegangen Eingabeform einzugeben. Der gesamte Musterfragebogen kann im Anhang B eingesehen werden.

6.1.3 Vergleichsfragebogen

Der Vergleichsfragebogen enthält dreizehn Fragen zur Person sowie zu der implementierten *SmartCrowd (Wear)*- Anwendung. Dazu wird dem Probanden zuerst die Frage über deren Händigkeit gestellt. Anschließend folgen zwei Fragen zur Bewertung der implementierten Eingabeformen für *Multiple Choice* und *Single Choice* Fragen sowie den Skalen. Im Anschluss daran, werden dem Probanden Fragen bezüglich dessen Umgang mit Gesundheitsdaten auf mobilen Endgeräten gestellt. Am Ende des Vergleichsfragebogens hat der Proband die Möglichkeit, weitere Anmerkungen oder Kritik in einem offenen Feld einzutragen. Der Vergleichsfragebogen kann im Anhang B eingesehen werden. Im nächsten Abschnitt wird die Durchführung der Studie beschrieben.

6.2 Durchführung

Bei der Durchführung der Studie ist es wichtig, dass alle Probanden ein möglichst ähnliches Szenario mit denselben Umweltbedingungen vorfinden. Das reduziert das Auftreten von verfälschten Werten und vereinfacht den Vergleich der ermittelten Daten. Aus diesem Grund findet für jeden Probanden die Studie in einem Seminarraum der Universität Ulm statt. Für die Durchführung der Studie werden als *Smartwatch* die *Motorola Moto 360 1. Generation* mit *Android* 5.1 und als *Smartphone* ein *Samsung Galaxy S4* mit der *Android* Version 5.0.1 verwendet. Dabei sind der Seminarraum und die verwendeten mobilen Geräte jeweils als unabhängige Variable der Studie anzusehen. Die Studie wird sitzend an einem Tisch durchgeführt. Eine Videokamera

zeichnet den Probanden auf, während er die Fragebögen auf der *Smartwatch* und auf dem *Smartphone* beantwortet.

6.2.1 Konstitution der Stichprobe

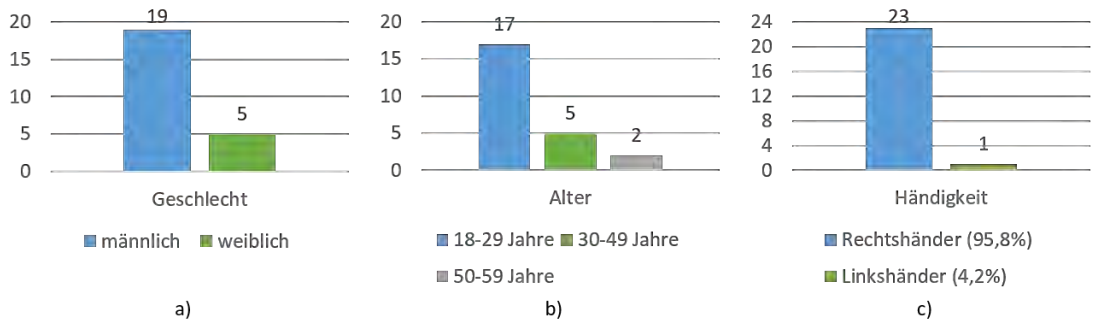

Abbildung 6.2: Konstitution der Stichprobe, anhand von a) Geschlecht, b) Alter und c) Händigkeit

Wie in Abbildung 6.2 a) und b) zu sehen ist, haben an der Studie fünf weibliche und neunzehn männliche Probanden teilgenommen. Die Probanden wurden in drei Altersgruppen von 18-29 Jahre, 30-49 Jahre und 50-59 Jahre gegliedert. Der größte Anteil der Probanden ordnet sich dabei in der Altersgruppe 18-29 Jahre ein, was an der großen Zahl an studentischen Probanden liegt. Wie in Abbildung 6.2 c) zu sehen, finden sich unter den 24 Probanden insgesamt 23 Rechtshänder und ein Linkshänder.

Die Abbildung 6.3 a) zeigt das Ergebnis zur Frage nach der Technikbegeisterung der Probanden. Diese Frage haben 79,2% der Probanden mit *ja* beantwortet. Von diesen 79,2% der Probanden sind 5,2% weiblich und 94,8% männlich. Kaum verwunderlich ist, dass von den 24 Probanden alle ein *Smartphone* besitzen. Bei der Verteilung der mobilen Betriebssysteme, die die Probanden verwenden, zeigt sich der deutlich Vorsprung bei den Marktanteilen, den *Android* derzeit in Deutschland hat. 66,7% der Probanden nutzen das Betriebssystem *Android* auf ihrem *Smartphone*. Das Betriebssystem von *Apple* wird hingegen von 29,2% der Probanden verwendet. Nur ein Proband (4,1%) nutzt das von *Microsoft* vertriebene *Windows Phone*. In Abbildung 6.3 b) ist das nochmals in einem Diagramm graphisch dargestellt.

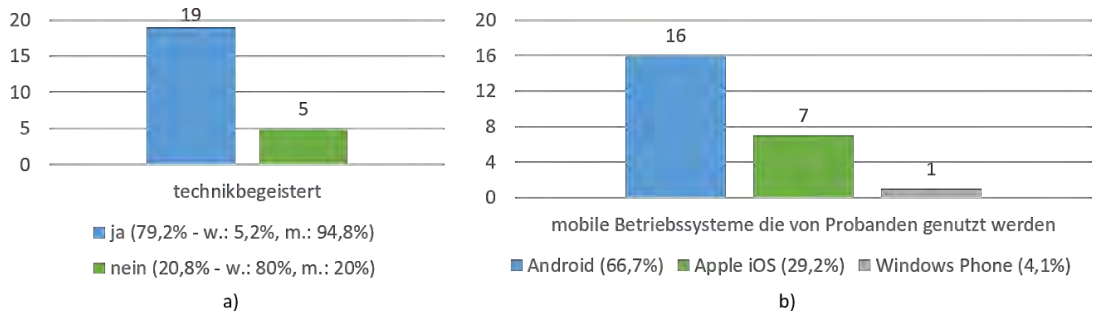

Abbildung 6.3: Konstitution der Stichprobe, anhand von a) der Technikbegeisterung und b) genutzten mobilen Betriebssystemen

Die Abbildungen 6.4 a) bis c) zeigt die Ergebnisse zu den Fragen bezüglich dem Besitz von Armbanduhren und *Smartwatches*. Die Frage, ob sie eine Armbanduhr besitzen, haben insgesamt 16 Probanden mit *ja* beantwortet. Interessant ist, dass jede der fünf weiblichen Probanden angibt eine Armbanduhr zu besitzen, aber nur zwei männliche Probanden eine *Smartwatch* nutzen. Betrachtet man die Kaufabsicht von den Probanden die noch keine *Smartwatch* besitzen, so möchte sich nur ein männlicher Proband in naher Zukunft eine solche zulegen.

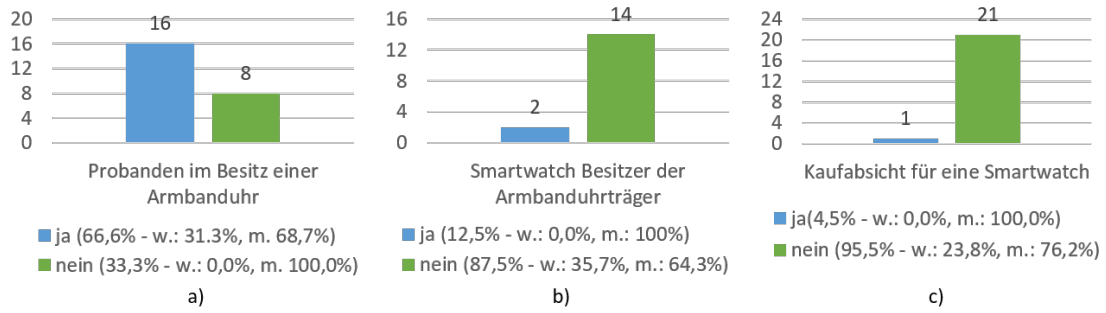

Abbildung 6.4: Konstitution der Stichprobe: a) Probanden im Besitz einer Armbanduhr, b) Besitzer einer *Smartwatch*, c) Kaufabsicht für eine *Smartwatch*

Als nächstes wurden die Probanden gefragt, ob sie sich vorstellen könnten Umfragen auf dem *Smartphone* oder der *Smartwatch* zu beantworten. Die Abbildung 6.5 zeigt die Ergebnisse. 19 Probanden (79,2%) können sich vorstellen in regelmäßigen Abständen Umfragen auf einem *Smartphone* auszufüllen. Die Mehrzahl der befragten Probanden

kann sich auch vorstellen, Umfragen auf einer *Smartwatch* mit einer entsprechenden Anwendung zu bearbeiten. Es können sich 15 Probanden (62,5%) vorstellen, Umfragen regelmäßig auf einer *Smartwatch* auszufüllen. Wie an den beiden Diagrammen in Abbildung 6.5 a) und b) zu erkennen ist, bevorzugt die Mehrheit der Probanden das *Smartphone* als Medium für die Teilnahme an Umfragen. Das Ergebnis der Frage zur Präferenz von *Smartphone* oder *Smartwatch* für die Bearbeitung von Umfragen in Abbildung 6.5 c) verdeutlicht dies nochmal. 22 Probanden (91,6%) haben angegeben, dass sie das *Smartphone* zur Beantwortung von Fragebögen der *Smartwatch* vorziehen würden.

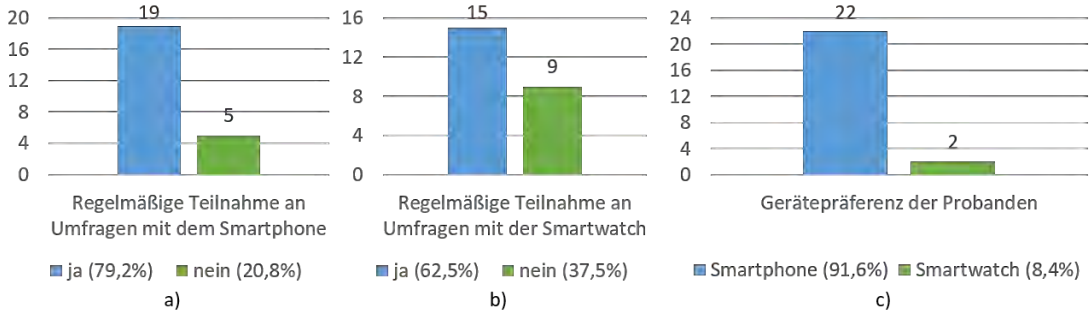

Abbildung 6.5: Regelmäßige Teilnahme an Umfragen: a) mit einem *Smartphone*, b) mit einer *Smartwatch* und c) Gerätepräferenz der Probanden

Die Probanden wurden zudem befragt, welche Anzahl an Fragen eine Umfrage maximal haben darf, damit sie von ihnen vollständig auf der *Smartwatch* bearbeitet wird. Die Abbildung 6.6 a) zeigt das Ergebnis. 8,3% der Probanden würden mehr als 15 Fragen auf der *Smartwatch* bearbeiten. 7-9 Fragen und 10-15 Fragen würden jeweils 29,2% der Probanden bearbeiten. 20,8% der Probanden finden 4-6 Fragen pro Umfrage angemessen und für 4,2% sind nur bis zu drei Fragen denkbar. Zwei Probanden (8,3%) wollten keine Angaben zu dieser Frage machen. Bei der Frage wie anstrengend die Probanden das Ausfüllen von Umfragen auf der *Smartwatch* empfinden, geben 20,8% der Probanden an, dass es für sie überhaupt nicht anstrengend ist. 45,8% der Probanden sagen, dass es ein wenig anstrengt. 29,2% empfinden das Bearbeiten von Umfragen auf *Smartwatches* mittelmäßig anstrengend und 4,2% ziemlich anstrengend. Betrachtet man diese Daten in Abbildung 6.6 b) im Diagramm ist festzuhalten, dass die Mehrheit

der befragten Probanden es für mittelmäßig bis wenig anstrengend findet, Umfragen auf einer *Smartwatch* zu bearbeiten.

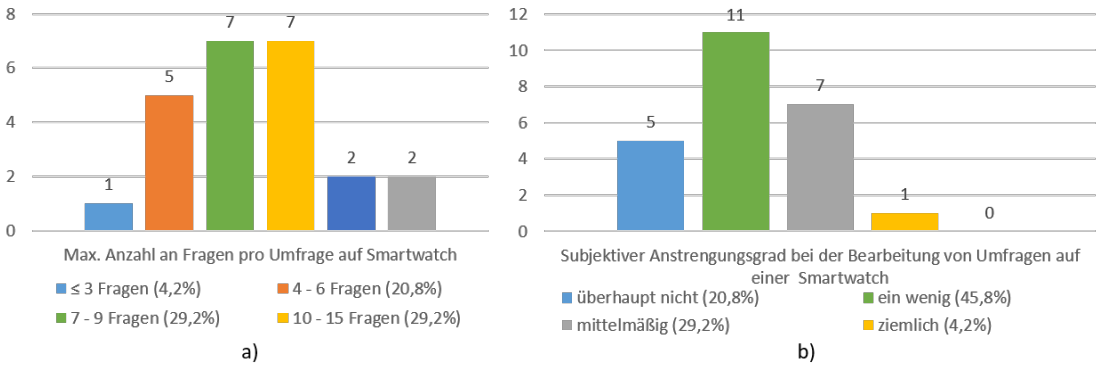

Abbildung 6.6: a) Max. Anzahl an Fragen pro Umfrage auf der *Smartwatch* b) Subjektiver Anstrengungsgrad bei der Bearbeitung von Umfragen auf einer *Smartwatch*

Im Weiteren wurden die Probanden zum Umgang mit ihren eigenen Gesundheitsdaten befragt. Bei der Frage bezüglich der Verwendung der Spracheingabe bei Gesundheitsfragen in öffentlichen Verkehrsmitteln, haben wie in Abbildung 6.7 a) zu erkennen, 91,6% der Probanden die Verwendung abgelehnt. Aus diesem Ergebnis ist zu entnehmen, dass bei der Abfrage von Gesundheitsdaten einer Person auf die Spracheingabe verzichtet werden sollte.

Neben der Frage zur Spracherkennung wurden die Probanden auch gefragt, wie sie sich fühlen würden, wenn sie über das *Smartphone* oder die *Smartwatch* Fragen über ihre Gesundheit beantworten sollen. Auch hier ist das Ergebnis deutlich. Auf dem *Smartphone* würden sich 25,0% der Probanden überhaupt nicht wohl fühlen. Weitere 25,0% würden sich ein wenig unwohl fühlen. 41.6% haben hierzu eine neutrale Meinung und 8,3% würden sich dabei ziemlich wohl fühlen. Betrachtet man die Abbildungen 6.7 b) und c) ist zu erkennen, dass sich die Gefühlslage der Probanden auf der *Smartwatch* nur minimal gegenüber der auf dem *Smartphone* verschlechtert. Trotzdem ist anzumerken, dass sich die Mehrheit der Probanden, bei der Beantwortung von Gesundheitsfragen auf einem *Smartphone* oder einer *Smartwatch* mittelmäßig bis überhaupt nicht wohlfühlt.

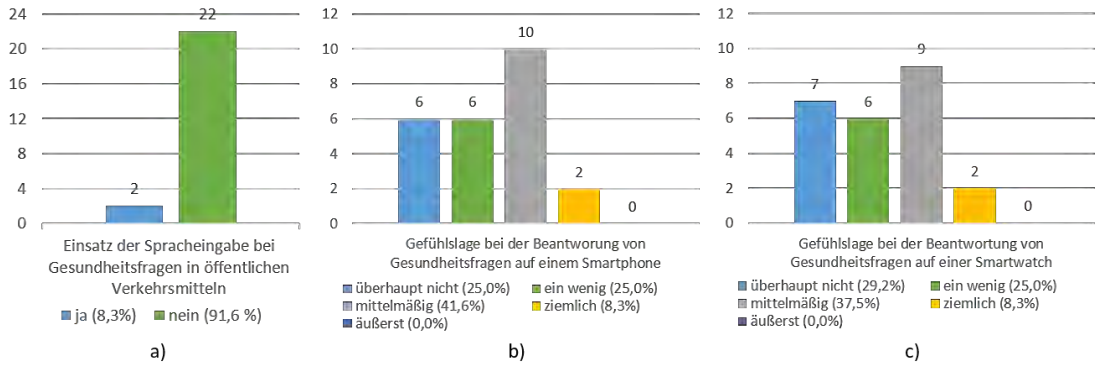

Abbildung 6.7: a) Einsatz der Spracheingabe bei Gesundheitsfragen, Gefühlslage bei der Beantwortung von Gesundheitsfragen auf b) *Smartphone* und c) *Smartwatch*

6.3 Ergebnisse

Mit der Studie sollen neben der Einstellung der Probanden gegenüber *Smartwatches* und Umfragen auch die entwickelten Eingabeformen auf der *Smartwatch* evaluiert werden. Im nachfolgenden Abschnitt werden die implementierten Eingabeformen anhand subjektiven Meinung der Probanden und der gemessenen Daten verglichen. Abschließend wird auf das gespeicherte Videomaterial eingegangen.

6.3.1 Vergleich der Eingabeformen Cards und List

In diesem Abschnitt werden die implementierten Eingabeformen miteinander verglichen und deren Vor- und Nachteile herausgearbeitet. Die Abbildung 6.8 a) zeigt die subjektive Meinung der Probanden zu den Eingabeformen *Cards* und *List* zur Beantwortung für *Multiple Choice* und *Single Choice* Fragen. Dabei präferieren 54,2% der Probanden die Eingabeform *List* und 41,6% die Eingabeform *Cards*. Ein Proband (4,2%) findet keine der beiden Eingabeformen sinnvoll. Am Ende der Studie merkten einige der Probanden an, dass Sie die Eingabeform *Cards* bei einer großen Anzahl an Antwortoptionen als sehr unübersichtlich empfinden, aber diese bei wenigen Antwortoptionen präferieren würden. Die Abbildung 6.8 b) zeigt die durchschnittliche Bearbeitungszeit einer Frage, die ein

Proband mit der entsprechenden Eingabeform auf einer Frage verbringt. Es ist deutlich zu erkennen, dass die Probanden mit Hilfe der Eingabeform *Cards* im Durchschnitt 0,65 Sekunden weniger Zeit für eine Frage aufbringen müssen. Anhand dieses Ergebnisses ist festzuhalten, dass die Mehrheit der Probanden die Eingabeform *List* bevorzugt. Rein objektiv betrachtet, sorgt die Eingabeform *Cards* für einen deutlichen Zeitgewinn von umgerechnet 10,4%.

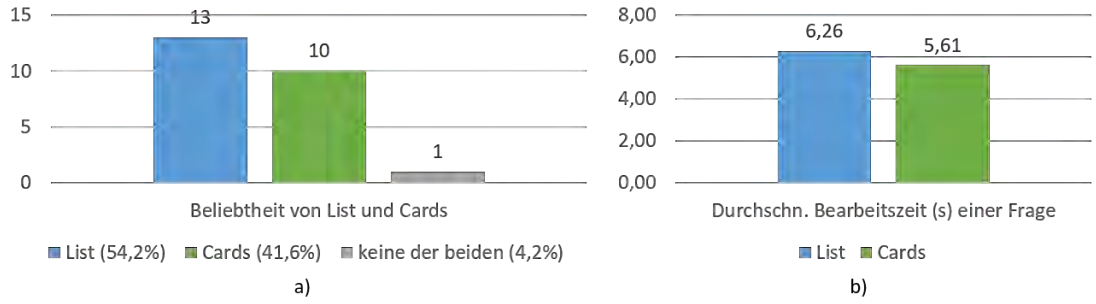

Abbildung 6.8: Beliebtheit der Eingabeformen *Cards* und *List* a) subjektiv durch Probanden und b) objektiv anhand von Messwerten

Im Weiteren wurden die beiden Eingabeformen *Cards* und *List* auf die Verwendung bei zwei, drei, fünf und sieben Antwortoptionen untersucht. Die Abbildung 6.9 stellt die Auswertung der ermittelten Daten in Diagrammen bereit. Im Schnitt braucht ein Proband 5,10 Sekunden, um eine Frage mit zwei Antwortoptionen mit der Eingabeform *Cards* zu bearbeiten. Bei drei Antwortoptionen sind es 5,65 Sekunden, bei fünf Antwortoptionen 6,87 Sekunden und bei sieben Antwortoptionen 7,30 Sekunden. Mit der Eingabeform *List* benötigen die Probanden im Schnitt bei zwei Antwortoptionen 5,52 Sekunden, bei drei Antwortoptionen 7,01 Sekunden, bei fünf Antwortoptionen 7,00 Sekunden und bei sieben Antwortoptionen 7,66 Sekunden. Daraus ist zu erkennen, dass die Probanden bei der Verwendung der Eingabeform *Cards* durchschnittlich eine kürzere Bearbeitungszeit für eine Frage haben. Bei der Eingabeform *List* ist anzumerken, dass sich die durchschnittliche Bearbeitungszeit bei drei und fünf Antwortoptionen kaum verändert. Betrachtet man die in Abbildung 6.9 e) dargestellte Gesamtübersicht, ist zudem festzustellen, dass mit zunehmender Anzahl von Antwortoptionen die Bearbeitungszeit einer

Frage ansteigt. Dies kann darauf zurückgeführt werden, dass bei vielen Antwortoptionen öfters zwischen *Pages* gewechselt oder gescrollt werden muss.

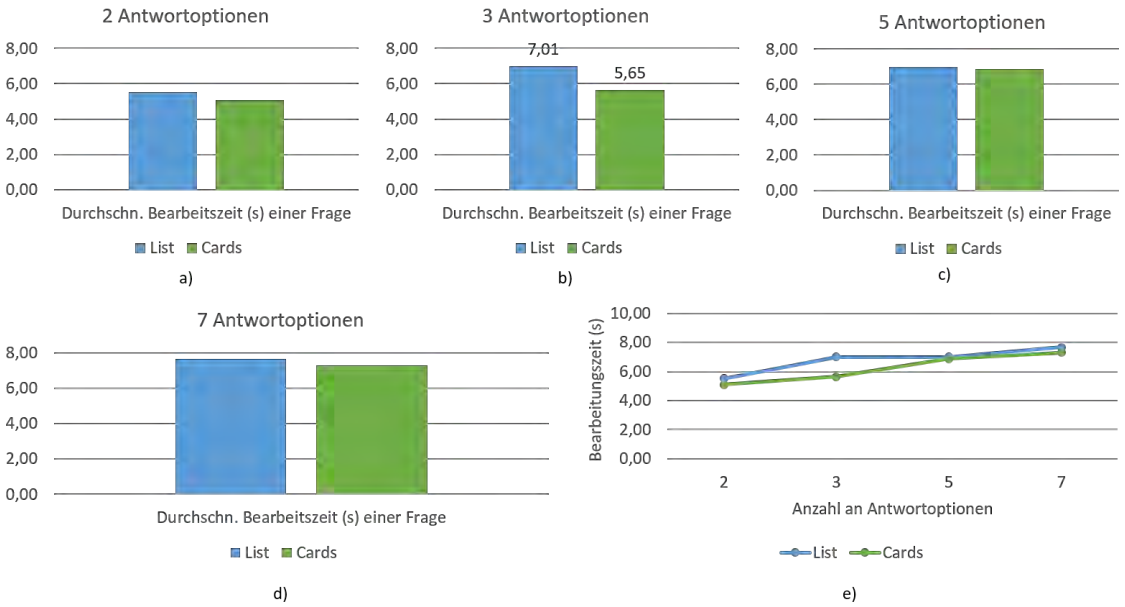

Abbildung 6.9: Durchschnittliche Bearbeitungszeit von *Single Choice* Fragen mit *List* und *Cards* bei a) 2 Antwortoptionen b) 3 Antwortoptionen c) 5 Antwortoptionen und d) 7 Antwortoptionen. e) Gesamtübersicht als Liniendiagramm

6.3.2 Vergleich der Eingabeformen LockScale und ButtonScale

Beim Vergleich der beiden Eingabeformen *ButtonScale* und *LockScale* für die Beantwortung von Fragen mit Skalen ist, wie in Abbildung 6.10 a) zu sehen, die Aussage der Probanden eindeutiger. Hier haben sich 79,2% der Probanden für die Eingabeform *ButtonScale* ausgesprochen und lediglich 12,5% für die *LockScale*. 8,3% finden keine der beiden Eingabeformen für die Auswahl von Werten auf einer Skala sinnvoll. 14 der 24 Probanden merkten an, dass das selektieren von genauen Werten auf einer Skala über die Eingabeform *LockScale* herausfordernd bis kaum möglich ist. Bei der Eingabeform *ButtonScale* wünschen sich die Probanden für die *Spull*-Funktion eine geringere und feste Sprungweite. Die implementierte beschleunigende Form sei schlecht kontrollierbar. Betrachtet man die gemessenen Werte der Bearbeitungszeit einer Frage,

bestätigen diese, wie in Abbildung 6.10 b) zu sehen ist, die subjektive Meinung der Probanden. Die durchschnittliche Bearbeitungszeit einer Frage beträgt bei der Eingabeform *ButtonScale* 7,47 Sekunden und bei der *LockScale* 11,59 Sekunden. Mit der Eingabeform *ButtonScale* sind die Probanden im Durschnitt um 35,5% schneller als mit der *LockScale*.

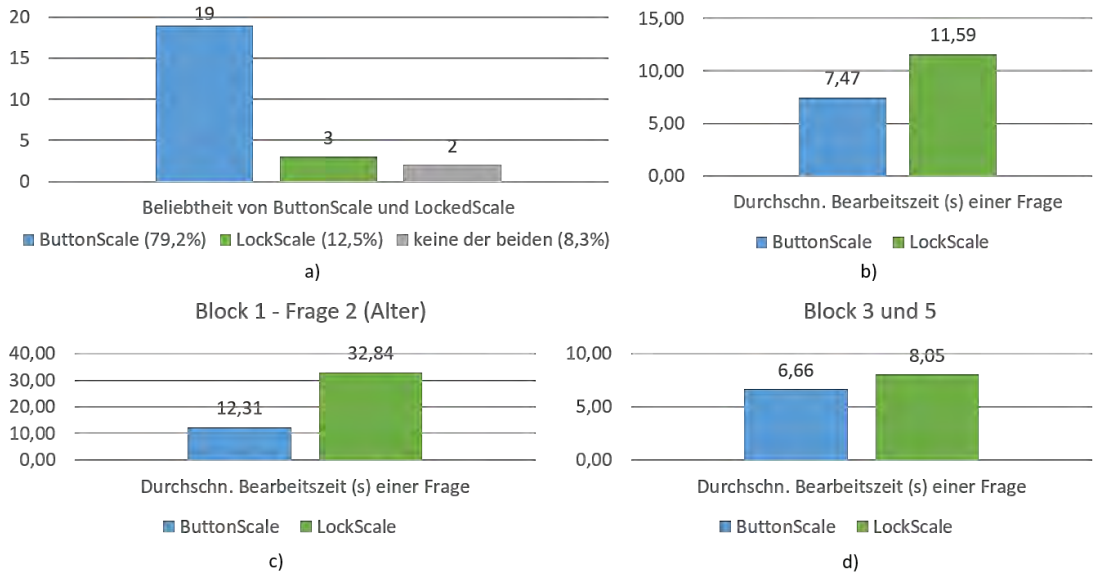

Abbildung 6.10: Priorisierung der Eingabeformen *ButtonScale* und *LockScale* a) subjektiv durch Probanden und b) objektiv anhand von Messwerten. Vergleich zwischen *ButtonScale* und *LockScale* bei der Eingabe von c) exakten Werten und d) Tendenzen

Der Musterfragen enthält eine Frage bei der die Probanden ihr Alter über die Eingabeformen *ButtonScale* und *LockScale* exakt eingeben müssen. In Abbildung 6.10 c) zu sehen, haben die Probanden mit der Eingabeform *LockScale* im Durchschnitt 32,84 Sekunden benötigt. Mit Hilfe der *ButtonScale* dauert es im Schnitt nur 12,31 Sekunden, bis die Frage beantwortet ist. Angesichts eines Zeitvorteils von 65,5% durch die *ButtonScale* ist festzustellen, dass die *LockScale* für die Eingabe von genauen Werten auf der *Smartwatch* ungeeignet ist. Rechnet man diese Frage aus dem vorhergehenden Vergleich in Abbildung 6.10 b) heraus, so gelangt man zu dem Ergebnis, dass die Probanden mit der *ButtonScale* bei der Eingabe von Tendenzen noch immer etwas schneller sind.

Der Abstand zwischen den Eingabeformen ist jedoch geringer. Der zeitliche Vorteil der *ButtonScale* zur *LockScale* beträgt dann noch 24,7%. Die Abbildung 6.10 d) zeigt diese graphisch. Differenziert man zwischen den Bearbeitungszeiten von drei, fünf und sieben stufigen Fragen, erhält man die in Abbildung 6.11 a) bis c) dargestellten Ergebnisse. Mit der Eingabeform *ButtonScale* benötigen die Probanden im Schnitt 8,15 Sekunden für dreistufige Fragen, 6,15 Sekunden für fünfstufige Fragen und für siebenstufige Fragen werden 6,53 Sekunden benötigt. Betrachtet man die *LockScale*, benötigen die Probanden 8,84 Sekunden für dreistufige Fragen, 7,49 Sekunden für fünfstufige Fragen und 7,82 Sekunden für siebenstufige Fragen. Dabei ist eindeutig festzustellen, dass in allen Kategorien die Probanden mit der *ButtonScale* die Fragen im Durchschnitt schneller beantwortet haben. Die Abbildung 6.11 d) zeigt die durchschnittliche Bearbeitungszeit von drei, fünf und sieben stufigen Fragen nochmal in einem Liniendiagramm. Interessant ist, dass mit zunehmender Anzahl an Stufen innerhalb einer Skala die Bearbeitungszeit abnimmt.

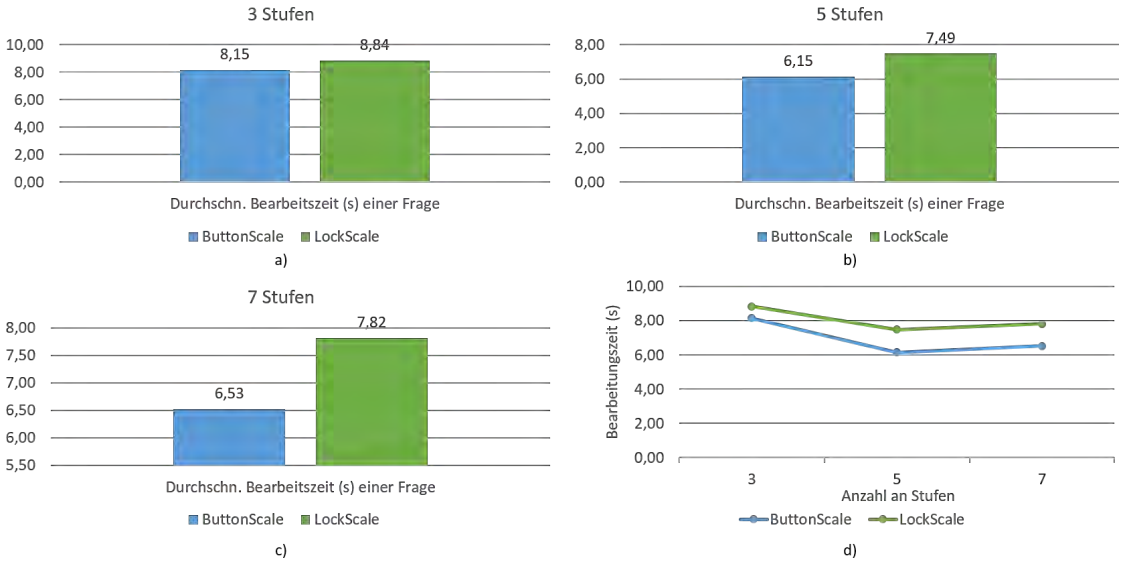

Abbildung 6.11: Durchschnittliche Bearbeitungszeit (s) auf Fragen mit den Eingabeformen *LockScale* und *ButtonScale* differenziert nach a) drei, b) fünf und c) sieben stufigen Fragen. d) Gesamtübersicht als Liniendiagramm

6.3.3 Smartwatch-Spracherkennung und Smartphone-Tastatur

Neben dem Vergleich der Eingabeformen sollte auch die Google Spracherkennung auf ihren Einsatz in einem Umfragesystem überprüft werden. Die Ergebnisse zeigen, dass 86,81% von den insgesamt 285 Spracheingaben von den Probanden sofort richtig eingegeben wurden. In 26 Fällen (9,12%) mussten sich die Probanden selbst korrigieren. Bei 1,05% der Fälle hat der Proband die Eingabe korrigiert und am Ende doch falsch eingegeben. 3,16% der Probanden haben ihre falsche Eingabe nicht korrigiert. Es haben 5 Probanden nach der Studie angemerkt, dass sie von der guten Spracherkennung überrascht waren. Dabei haben sie die Spracherkennung mit sehr gut bis annähernd perfekt beschrieben. Die Ursachen der Korrekturen und der falschen Eingaben kann nur schwer nachvollzogen werden. Aus subjektiver Sicht ist aber festzustellen, dass die überwiegende Anzahl der nötigen Korrekturen von den Probanden selbst herbeigeführt wurden. Auch die fehlerhaften Eingaben sind überwiegend der falschen Eingabe durch die Probanden zuzuordnen. Aus diesem Grund ist anzunehmen, dass die Genauigkeit der Spracherkennung über den 86,67% liegt. Vergleicht man die Spracheingabe auf der *Smartwatch* und die *Smartphone*-Tastatur anhand ihrer Eingabegeschwindigkeit, ist die Spracheingabe auf der *Smartwatch* im Vorteil. So benötigten die Probanden auf der *Smartwatch* im Durchschnitt 0,21 Sekunden pro Zeichen. Mit der *Smartphone*-Tastatur hingegen benötigten sie im Schnitt 0,62 Sekunden. Somit ist anzumerken, dass bei der Bearbeitung von offenen Fragen auf der *Smartwatch* dem Nutzer keine zeitlichen Nachteile entstehen.

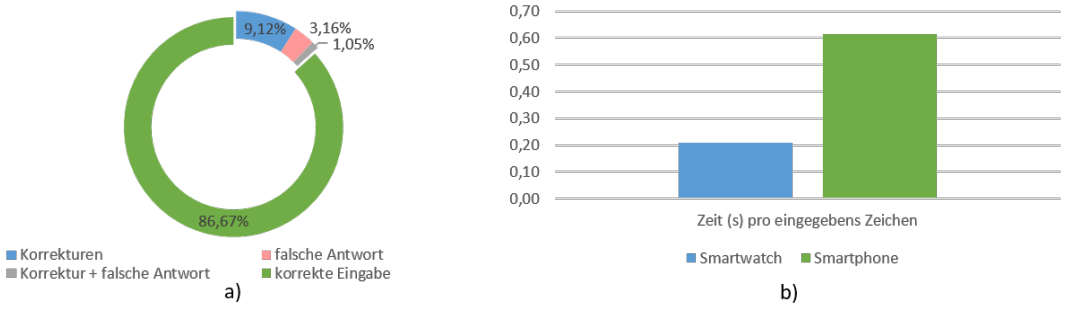

Abbildung 6.12: Spracheingabe: a) Statistik über korrekte und falsche Eingaben b) Eingabegeschwindigkeit der Spracherkennung und der *Smartphone*-Tastatur

6.3.4 Fortschrittsanzeige

Zudem wurden die Probanden gefragt, ob sie die implementierte Fortschrittsanzeige während der Studie wahrgenommen haben und ob sie diese als hilfreich erachten. Die Abbildung 6.13 zeigt das Ergebnis. 19 Probanden (79,2%) haben angegeben, dass sie die Fortschrittsanzeige als solche wahrgenommen haben. Lediglich 5 Probanden (10,2%) haben diese während der Studie nicht wahrgenommen. Wie in Abbildung 6.13 b) zu sehen ist, befindet die Mehrheit der 19 Probanden die Fortschrittsanzeige für nützlich. Drei Probanden merkten an, dass die Fortschrittsanzeige deutlicher hervorgehoben werden sollte, damit sie besser zu erkennen ist.

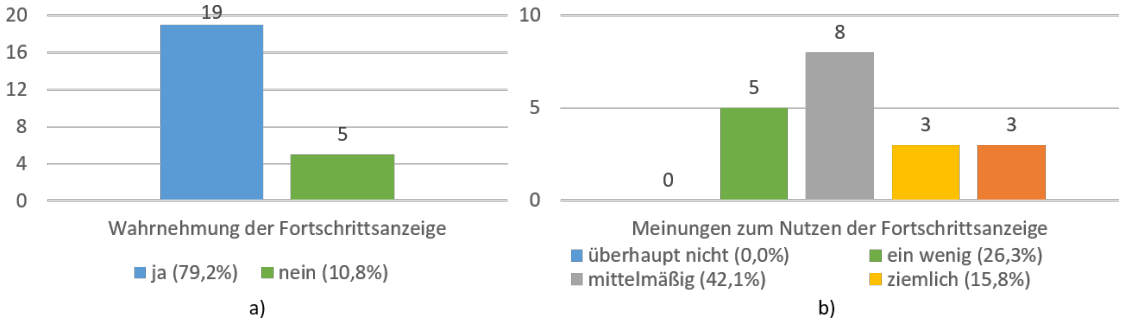

Abbildung 6.13: Fortschrittsanzeige a) Wahrnehmung und b) Brauchbarkeit für den Benutzer

6.3.5 Gesamtzeiten der Eingabeformversionen

Die Abbildung 6.14 a) zeigt die durchschnittlichen Zeiten, die von den Probanden benötigt wird um den Musterfragebogen mit den unterschiedlichen Eingabeformversionen zu bearbeiten. Es sticht heraus, dass die Bearbeitungszeit des gesamten Fragebogens auf dem *Smartphone* (V5) mit 198,63 Sekunden die Kürzeste ist. Schaut man auf die Ergebnisse der *Smartwatch* (V1-V4), so ist erkennbar dass die Version 1 (V1) mit 203,13 Sekunden die schnellste Version auf der *Smartwatch* darstellt. Betrachtet man jedoch die in Abbildung 6.14 b) gezeigten *Box-Whisker-Plots* wird deutlich, dass die Stichprobe eine starke Streuung im Bezug auf die benötigten Zeiten aufweist. Es ist auch zu sehen, dass die Eingabeformversion V3 die größte Streuung aufweist. Vergleicht man den

Median der einzelnen Eingabeformen miteinander, so liegt die Eingabeformversion 3 (V3) mit den Eingabeformen *Cards* und *ButtonScale* deutlich vor Version 1 (V1) und dem *Smartphone* (V5). Aus diesem Grund ist an dieser Stelle der Median aussagekräftiger als das arithmetische Mittel. Für die weitere Auswertung und Beurteilung der Eingabeformen sollte deshalb der Median herangezogen werden.

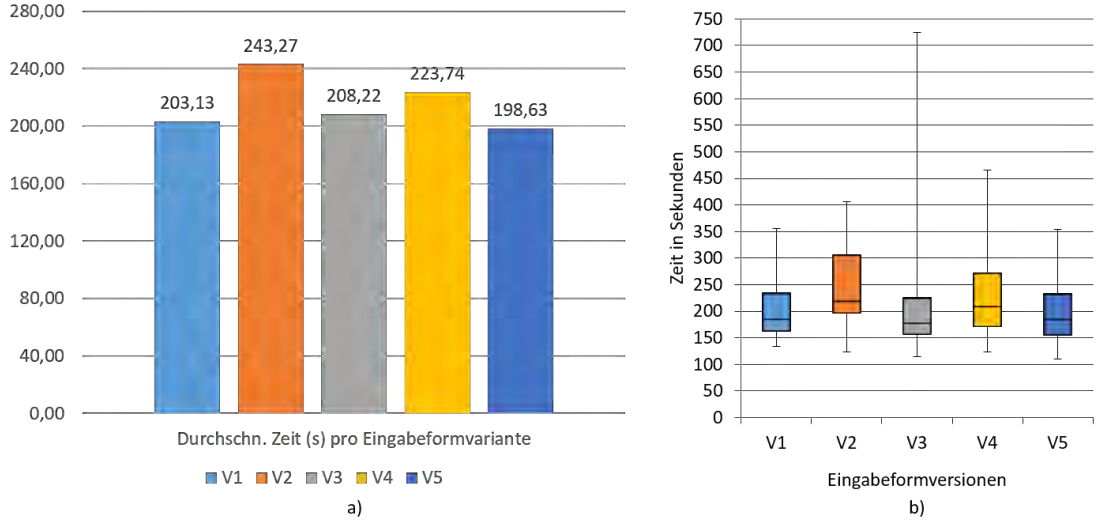

Abbildung 6.14: Bearbeitungszeit des Musterfragebogens: a) durchschn. Zeit (s) und b) *Box-Whisker-Plot* Darstellung

6.3.6 Verhalten der Probanden während der Studie

Wie bereits beschreiben, wurden die Probanden während der Durchführung der Studie mit einer Videokamera aufgezeichnet. Im diesem Abschnitt werden die Beobachtungen, die bei der Sichtung des Videomaterials gemacht worden sind, beschrieben. Die erste Beobachtung die gemacht wurde ist, dass alle Probanden ihren Arm mit der *Smartwatch* auf dem Tisch ablegen. Selbst wenn sich die Probanden zurück lehnen, wird die Hand mit der *Smartwatch* auf dem Tisch aufgelegt. Müssen die Probanden eine Frage per Spracheingabe beantworten, ist festzustellen, dass zu Beginn entweder der Arm mit der *Smartwatch* in Richtung Mund geführt oder der Kopf zu Uhr herab gesenkt wird. Nach der Spracheingabe wird der Arm sofort wieder vollständig auf dem Tisch aufgelegt.

Umso weiter die Studie fortschreitet, desto kleiner werden diese Bewegungen. Es ist davon auszugehen, dass die Probanden erkennen, dass die Spracheingabe auch auf eine größere Distanz funktioniert und dadurch diese Bewegungen nicht mehr für nötig erachten. Das Videomaterial zeigt auch, dass einige der Probanden bei der genauen Eingabe ihres Alters über die Eingabeform *LockScale* sichtlich angespannt wirken. Die meisten aber einen Ehrgeiz entwickeln, die Angabe ihres Alters korrekt einzugeben.

6.4 Auswertung

Wirft man einen Blick auf die im vorigen Abschnitt präsentierten Studienergebnisse einer *Smartwatch*, so kann die erste Hypothese bestätigt werden. *Smartwatches* können mittels einer optimierten *Smartwatch*-Anwendung in einem *Mobile Crowd Sensing System* eingesetzt werden. Die *SmartCrowd (Wear)*- Anwendung kann vor allem bei offenen Fragen mit der integrierten Spracheingabe punkten. Die objektiven Messwerte und die subjektive Meinung der Probanden zeigen jedoch, dass die Beantwortung von Fragen mittels der implementierten *LockScale* und *ButtonScale* nicht optimal funktioniert. Ein weiterer Punkt ist, dass bei *Multiple Choice* und *Single Choice* Fragen mit mehreren Antwortoptionen nicht alle Antworten sofort für den Benutzer sichtbar sind. Somit wird die implementierte Anwendung auf dem *Smartphone* von den Probanden subjektiv bevorzugt und als benutzerfreundlicher empfunden. Mit der entwickelten *SmartCrowd (Wear)*- Anwendung kann aus technischen Gründen zwar keine Anwendung bereitgestellt werden, die ein *Smartphone* in einem *Mobile Crowd Sensing System* überflüssig macht. Sie kann aber unterstützend für Benachrichtigungen bis hin zur Bearbeitung von ganzen Umfragen herangezogen werden.

Die zweite Hypothese wird durch die Studie nicht bestätigt. Die größte Anzahl an Probanden hat bei der Befragung angegeben, dass sie es mittelmäßig bis überhaupt nicht anstrengend finden, Umfragen auf der *Smartwatch* auszufüllen. Auch auf den Videoaufzeichnungen konnte kein Erschöpfungszustand festgestellt werden.

Durch die Ergebnisse des Vergleichsfragebogens konnte die dritte Hypothese bestätigt werden. Die Mehrheit der Probanden bevorzugt das *Smartphone* vor der *Smartwatch*

zum Bearbeiten von Umfragen. Auch die Ergebnisse zur regelmäßigen Bearbeitung von Umfragen auf dem *Smartphone* oder der *Smartwatch* untermauern diese Aussage.

Die vierte Hypothese konnte nicht bestätigt werden. Nur fünf Probanden haben sich für vier bis sechs Fragen ausgesprochen. Die Mehrzahl der Probanden ist aber der Meinung, dass eine Umfrage auch mehr Fragen beinhalten kann. Dabei liegt nach subjektiver Meinung der Probanden die Obergrenze für die Anzahl der Fragen in einer Umfrage für *Smartwatches* bei 15 Fragen.

Im Weiteren wurde behauptet, dass die Mehrheit der Benutzer keine Fragen zur ihrer Gesundheit in öffentlichen Verkehrsmitteln per Spracheingabe beantworten wollen. Diese fünfte Hypothese konnte wiederum bestätigt werden. Eine absolute Mehrheit von 91,6% der Probanden findet, dass an dieser Stelle eine andere Lösung gefunden werden muss.

Auch die sechste Hypothese konnte bestätigt werden. Die Mehrheit der Probanden fühlt sich bei der der Beantwortung von Fragen zu ihrer Gesundheit auf elektronischen Geräten wie *Smartphone* und *Smartwatch* eher unwohl.

Die siebte Hypothese konnte ebenfalls bestätigt werden. Nur wenige der Probanden haben die Fortschrittsanzeige nicht als solche wahrgenommen. Die überwiegende Anzahl der Probanden war sich einig, dass eine solche Funktion sinnvoll ist.

Bei der Verwendung der Spracherkennung zur Beantwortung von offenen Fragen wurde eine deutliche Mehrzahl der Spracheingaben sofort richtig erkannt. Nur in wenigen Fällen musste der Benutzer seine Eingabe korrigieren. Viele Probanden gaben nach der Studie im Vergleichsfragebogen an, dass sie von der guten Spracherkennung beeindruckt sind. Die achte Hypothese kann somit bestätigt werden.

Beim Vergleich der beiden Eingabeformen *List* und *Cards* haben sich die Probanden mit knapper Mehrheit subjektiv für die Eingabeform *List* entschiedenen. Betrachtet man die Eingabeformen objektiv anhand der gemessenen Werte, so ist die Eingabeform *Cards* gegenüber der Eingabeform *List* eindeutig im Vorteil. Außerdem zeigen die gemessenen Werte ebenso, dass die Eingabeform *List* mit zunehmender Anzahl an Antwortoptionen in der Geschwindigkeit zu der Eingabeform *Cards* aufholen kann. Darüber hinaus müssen die technischen Einschränkungen der beiden Eingabeformen berücksichtigt werden. Die Eingabeform *List* ist lediglich für Antwortoptionen mit kurzen Antworttexten

geeignet. *Cards* hingegen sind bei einer großen Anzahl an Antwortoptionen unübersichtlich. Aus diesem Grund kann an dieser Stelle kein eindeutiger Sieger zwischen den Eingabeformen ermittelt werden. Die neunte Hypothese kann somit nicht bestätigt werden.

Der Vergleich zwischen den beiden Eingabeformen *ButtonScale* und *LockScale* zeigt, dass beide Eingabeformen Schwächen haben und dass eine Skala auf der *Smartwatch* nicht die beste Eingabeform ist. Aus subjektiver Sicht der Probanden wird die *ButtonScale* bevorzugt. Sie sei zur Selektierung genauer Werte besser geeignet. Dies wird objektiv von der gemessenen Werten so auch bestätigt. Infolgedessen kann die zehnte Hypothese wiederum bestätigt werden.

Abschließend wird die Bearbeitungszeit der Eingabeformen in Abhängigkeit von der Anzahl der Antwortoptionen bzw. Stufen betrachtet. Bei den Eingabeformen *Cards* und *List* steigt die Bearbeitungszeit mit zunehmender Anzahl an Antwortoptionen. Betrachtet man die Eingabeformen *ButtonScale* und *LockScale*, sinkt dort die Bearbeitungszeit mit zunehmender Anzahl der Stufen. Aus diesem Grund kann die Hypothese nur für die Eingabeformen *Cards* und *List* bestätigt werden. Für die Eingabeformen *LockScale* und *ButtonScale* bestätigt sich diese jedoch nicht.

7

Zusammenfassung und Ausblick

Nachdem in den vorigen Kapiteln die theoretischen und praktischen Aspekte sowie die durchgeführte Studie vorgestellt wurden, sollen diese nun zusammengeführt und abschließend bewertet werden. Im anschließenden Abschnitt 7.1 wird der Inhalt dieser Arbeit zusammengefasst. Abschließend wird im Abschnitt 7.2 ein Ausblick über die zukünftige Weiterentwicklung gegeben.

7.1 Zusammenfassung

Das Ziel dieser Arbeit war es, *Smartwatches* auf ihre Tauglichkeit für den Einsatz in *Mobile Crowd Sensing Systemen* zu überprüfen. Hierzu sollte eine *Smartwatch*-Anwendung entwickelt und anschließend in einer Studie evaluiert werden. Dieses Ziel wurde mit der Implementierung eines Umfragesystems erreicht. Dabei wurden ein Webservice und zwei mobile Anwendungen für *Android-Smartphones* und *Android Wear-Smartwatches* entwickelt.

Die Arbeit zeigt, dass mit dem entwickelten Umfrage-Designkonzept für *Smartwatches* eine *Smartwatch*- Anwendung für den Einsatz in einem *Mobile Crowd Sensing System* realisiert werden kann. Für die Bearbeitung von Umfragen auf einer *Smartwatch* wurden unterschiedliche Eingabeformen für *Multiple Choice* und *Single Choice* Fragen sowie Skalen und offene Fragen entwickelt. Anschließend wurden diese, in einer Studie, auf deren Eignung für die entsprechenden Fragearten evaluiert.

Die Studienergebnisse zeigen, dass sich die Spracherkennung für die Beantwortung von offenen Fragen bestens eignet. Außerdem lässt sich durch diese eine deutlich bessere

Eingabegeschwindigkeit als mit der *Smartphone*-Tastatur erreichen. Beim Vergleich der beiden Eingabeformen *Cards* und *List* konnte keine der beiden als eindeutig bessere Alternative identifiziert werden. Durch die subjektive Einschätzung der Probanden lässt sich aber festhalten, dass die Eingabeform *List* für Fragen mit vielen Antwortoptionen aufgrund der Übersichtlichkeit, geeigneter erscheint. Zudem ist festzuhalten, dass mit zunehmender Anzahl an Antwortoptionen auch die Bearbeitungszeit einer Frage ansteigt.

Bei den beiden Eingabeformen *LockScale* und *ButtonScale* sind die Ergebnisse jedoch eindeutig. Die Eingabeform *LockScale* ist zur Selektierung von exakten Werten ungeeignet. Aber auch bei Tendenzfragen schnitt die *ButtonScale* besser ab. Es ist festzustellen, dass die Bearbeitungszeit von Fragen bei beiden Eingabeformen mit einer steigenden Zahl an Stufen sinkt. Das ist exakt der gegenteilige Effekt wie es bei den Eingabeformen *List* und *Cards* zu sehen ist. Daraus lässt sich ableiten, welche Eingabeform für welche Anzahl an Antwortoptionen besser geeignet ist. Die Eingabeformen *ButtonScale* und *LockScale* sollten daher für Fragen mit vielen Antwortoptionen eingesetzt werden. Die Eingabeformen *List* und *Cards* hingegen bei wenigen Antwortoptionen.

Die Studie zeigt auch, dass die Probanden bei Umfragen zu ihrer Gesundheit eher skeptisch gegenüber einem mobilen Umfragesystem sind. Beispielsweise würden nur zwei der 24 befragten Probanden, Fragen zu ihrer Gesundheit in öffentlichen Verkehrsmitteln per Spracheingabe beantworten. Die meisten der befragten Probanden hätten aber auch bei anderen Eingabeformen ein schlechtes Gefühl dabei.

Vergleicht man die durchschnittliche Bearbeitungszeit auf der *Smartwatch* ((203,13s + 243,27s + 208,22s + 223,74s + 198,63s) / 5 = 215,40s), mit der auf dem *Smartphone* (198,63s), benötigt man mit dem *Smartphone* im Durchschnitt nur 8,4% länger. Betrachtet man die im Durchschnitt schnellste Kombination der Eingabeversion V1(203,13s), benötigen die Probanden mit der *Smartwatch* nur 2,3% mehr Zeit. Anhand dieser Messungen kann die *Smartwatch* als Alternative zum *Smartphone* in *Mobile Crowd Sensing Systemen* angesehen werden.

Betrachtet man *Smartwatches* auf Basis von *Android Wear*, dann ist noch deutliches Entwicklungspotential zu erkennen. Zum Zeitpunkt dieser Arbeit unterliegen *Smartwat-*

ches erheblichen Einschränkungen. Unter *Android Wear* kann keine Internetverbindung hergestellt werden. *Smartwatches* die das *Android Wear* Betriebssystem nutzen, sind somit immer auf eine Verbindung zu einem *Smartphone* angewiesen. Für die entwickelte *Smartwatch*- Anwendung ist, zur Anbindung an den Webservice, die Entwicklung einer *Smartphone*- Anwendung unumgänglich. Auch die kleine Baugröße schränkt die Interaktion zwischen Benutzer und *Smartwatch* ein. Bei der vorgestellten Anwendung erfordert dies die Trennung von Frage und Antworten in separate *Pages*.

Fakt ist, dass lediglich zwei der 24 Probanden im Besitz einer *Smartwatch* waren und lediglich einer von den verbleibenden 22 Probanden sich in naher Zukunft eine solche zulegen möchte. Daraus lässt sich schließen, dass *Smartwatches* in ihrer bisherigen Form noch kein ausgereiftes Produkt darstellen. Ein erheblicher Mehrwert für den Kunden fehlt. Trotzdem können sich 15 von 24 Probanden vorstellen, Umfragen in regelmäßigen Abständen auf einer *Smartwatch* durchzuführen. Um genaue Aussagen, über die Akzeptanz von *Smartwatches* in Umfragesystemen treffen zu können, müsste jedoch erst eine Langzeitstudie durchgeführt werden.

7.2 Ausblick

Das in dieser Arbeit entwickelte Umfragesystem diente als Testsystem für die Durchführung der in Kapitel 6 beschriebenen Studie. In einem weiteren Schritt könnte die TrackYourTinnitus Plattform um die Unterstützung von *Smartwatches* erweitert werden. Die *TrackYourTinnitus* Plattform unterstützt die mobilen Betriebssysteme *Android* und *iOS*. Aus diesem Grund wäre eine Portierung der in dieser Arbeit entwickelten *Android Wear* Anwendung auf *Apple Watch OS* sinnvoll. So könnten alle Teilnehmer am *TrackYourTinnitus* Projekt von dieser Erweiterung profitieren. In diesem Zusammenhang wäre auch die Durchführung einer Langzeitstudie über die Akzeptanz von *Smartwatches* innerhalb des *TrackYourTinnitus* Projektes möglich.

Im Hinblick auf eine Erweiterung der *TrackYourTinnitus* Plattform, sind die integrierten Sensoren von *Smartwatches* von besonderem Interesse. Bislang konnten nur subjektive Einschätzungen der Patienten über Fragebögen ermittelt werden. In Zukunft könnten die-

se mit Daten von körpernahen Sensoren, wie beispielsweise dem Herzfrequenzsensor, ergänzt werden.

Das Institut für Datenbanken und Informationssysteme der Universität Ulm hat sich bereits mit dem Einsatz von externen Sensoren wie beispielsweise Herzfrequenzsensoren oder Oximetern in mobilen Anwendungen beschäftigt [34]. Bis heute waren solche medizinische Geräte Arztpraxen oder anderen medizinischen Einrichtungen vorenthalten. Durch die wachsende Verbreitung und die Weiterentwicklung von *Smartwatches* werden immer mehr Menschen solche Sensoren tagtäglich an sich tragen. Die Dokumentation und Analyse dieser Sensordaten in Verbindung mit dem bisherigen Umfragesystem, könnte weitere Erkenntnisse im Forschungsbereich Tinnitus ergeben.

Literaturverzeichnis

[1] BRODERSEN, B. : *WLAN-Sync für Smartwatches und bald auch Audio-Support fürs Telefonieren*. http://www.areamobile.de/news/32515-android-wear-update-5-1-wlan-sync-fuer-smartwatches-und-bald-auch-audio-support-fuers-telefonieren, Version: 21.03.2016

[2] DITZIG, J. : *Funktionsanalyse und Funktionserweiterung für eine personalisierte Mobile Crowd Sensing Plattform mit Algorithmen-Entwicklung*, Universität Ulm, Masterarbeit, Januar 2016

[3] FLOEMER, A. : *Android Wear: Googles Software-Plattform für Smartwatches und Co*. http://www.giga.de/downloads/android-wear/, Version: 21.03.2016

[4] GAMMA, E. ; HELM, R. ; JOHNSON, R. u. a.: *Design patterns: elements of reusable object-oriented software*. 37th printing. Addison-Wesley. – Online–Ressource (xv, 395 p.) S. http://proquest.tech.safaribooksonline.de/0201633612. – ISBN 0–201–63361–2

[5] GEIGER, P. ; SCHICKLER, M. ; PRYSS, R. ; SCHOBEL, J. ; REICHERT, M. : Location-based Mobile Augmented Reality Applications: Challenges, Examples, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 383–394

[6] http://www.softselect.de/business-software-glossar/backend

[7] GOOGLE, I. : *Android*. https://www.android.com/, Version: 21.03.2016

[8] GOOGLE, I. : *Android Activity*. http://developer.android.com/reference/android/app/Activity.html, Version: 21.03.2016

[9] GOOGLE, I. : *Android BroadcastReceiver*. http://developer.android.com/reference/android/content/BroadcastReceiver.html, Version: 21.03.2016

[10] GOOGLE, I. : *Android Fragments*. http://developer.android.com/guide/components/fragments.html, Version: 21.03.2016

[11] GOOGLE, I. : *Android Intents and Intent Filters*. http://developer.android.com/guide/components/intents-filters.html, Version: 21.03.2016

[12] GOOGLE, I. : *Android Services*. http://developer.android.com/guide/components/services.html, Version: 21.03.2016

[13] GOOGLE, I. : *Android Wear 2D Picker*. http://developer.android.com/training/wearables/ui/2d-picker.html, Version: 21.03.2016

[14] GOOGLE, I. : *Android Wear Cards*. http://developer.android.com/training/wearables/ui/cards.html, Version: 21.03.2016

[15] GOOGLE, I. : *Android Wear Defining Layouts*. http://developer.android.com/training/wearables/ui/layouts.html, Version: 21.03.2016

[16] GOOGLE, I. : *Android Wear Lists*. http://developer.android.com/training/wearables/ui/lists.html, Version: 21.03.2016

[17] GOOGLE, I. : *Android Wear Sending and Syncing Data*. http://developer.android.com/training/wearables/data-layer/index.html, Version: 21.03.2016

[18] GOOGLE, I. : *Creating Custom UIs for Wear Devices*. http://developer.android.com/training/wearables/ui/index.html, Version: 21.03.2016

[19] GOOGLE, I. : *Handling Data Layer Events*. http://developer.android.com/training/wearables/data-layer/events.html, Version: 21.03.2016

[20] GOOGLE, I. : *Packaging Wearable Apps*. http://developer.android.com/training/wearables/apps/packaging.html, Version: 21.03.2016

[21] GOOGLE, I. : *Sending and Receiving Messages*. http://developer.android.com/training/wearables/data-layer/messages.html, Version: 21.03.2016

[22] HÜBSCHER, H. ; RATHGEBER, C. ; RICHTER, K. ; PETERSEN, H. J. ; SCHARF, D. : *IT-Kompendium: Kompaktwissen Informationstechnik*. Westermann, 2011

[23] HERRMANN, J. : *Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten*, Universität Ulm, Masterarbeit, März 2014

[24] INTERNATIONAL, E. : The JSON Data Interchange Format. (2013)

[25] KÜNNETH, T. : *Android 4: Apps Entwickeln mit dem Android SDK*. 2. aktualisierte Auflage. Galileo Computing, 2012

[26] KOMPENDIUM, E. : *Bluetooth Low Energy / Bluetooth Smart (4.0 / 4.1 / 4.2)*. http://www.elektronik-kompendium.de/sites/kom/1805171.htm, Version: 28.03.2016

[27] MACKENZIE, I. S. ; CASTELLUCCI, S. J.: Empirical Research Methods for Human-Computer Interaction. (2009), April. http://www.yorku.ca/mack/CourseNotes.pdf

[28] MOLLER, A. R. ; LANGGUTH, B. ; HAJAK, G. ; KLEINJUNG, T. ; CACACE, A. : *Tinnitus: Pathophysiology and Treatment: Pathophysiology and Treatment*. Elsevier, 2007

[29] PETER M. KREUZER, B. L. Veronika Vielsmeier V. Veronika Vielsmeier: Chronischer Tinnitus – eine interdisziplinäre Herausforderung. In: *Deutsches Ärzteblatt* (2013), April

[30] PRYSS, R. ; REICHERT, M. ; LANGGUTH, B. ; SCHLEE, W. : Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy and Research. In: *IEEE 4th International Conference on Mobile Services (MS 2015)* (2015), June. http://dbis.eprints.uni-ulm.de/1152/

[31] RAGHU K. GANTI, H. L. Fan Ye Y. Fan Ye: Mobile Crowdsensing: Current State and Future Challenges. In: *IEEE Communications Magazine* (2011)

[32] SCHICKLER, M. ; PRYSS, R. ; SCHOBEL, J. ; REICHERT, M. : An Engine Enabling Location-based Mobile Augmented Reality Applications. Version: 2015. http://dbis.eprints.uni-ulm.de/1137/. In: *10th International Conference on Web Information Systems and Technologies (Revised Selected Papers)*. Springer (LNBIP 226), 363–378

[33] SCHICKLER, M. ; REICHERT, M. ; PRYSS, R. u. a.: *Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health*. Springer-Verlag, 2015

[34] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. u. a.: Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 509–518

[35] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; MAIER, F. ; REICHERT, M. : Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 371–382

[36] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; REICHERT, M. : Process-Driven Data Collection with Smart Mobile Devices. In: *10th International Conference on Web Information Systems and Technologies (Revised Selected Papers)*. Springer, 2015 (LNBIP 226), S. 347–362

[37] STATISTA: *Prognose: Anzahl der Smartphone-Nutzer weltweit von 2012 bis 2019*. `http://de.statista.com/statistik/daten/studie/309656/umfrage/prognose-zur-anzahl-der-smartphone-nutzer-weltweit/`, Version: 21.03.2016

[38] TILKOV, S. : *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien*. 1. Auflage. dpunkt Verlag, 2009

[39] TIM BERNERS-LEE, R. F. u. a.: *Hypertext Transfer Protocol – HTTP/1.1*. `https://www.ietf.org/rfc/rfc2616.txt`, Juni 1999. – Version: 21.03.2016

A

Studieneinführung

Abbildung A.1: Studieneinführung-Folien Teil 1

Abbildung A.2: Studieneinführung-Folien Teil 2

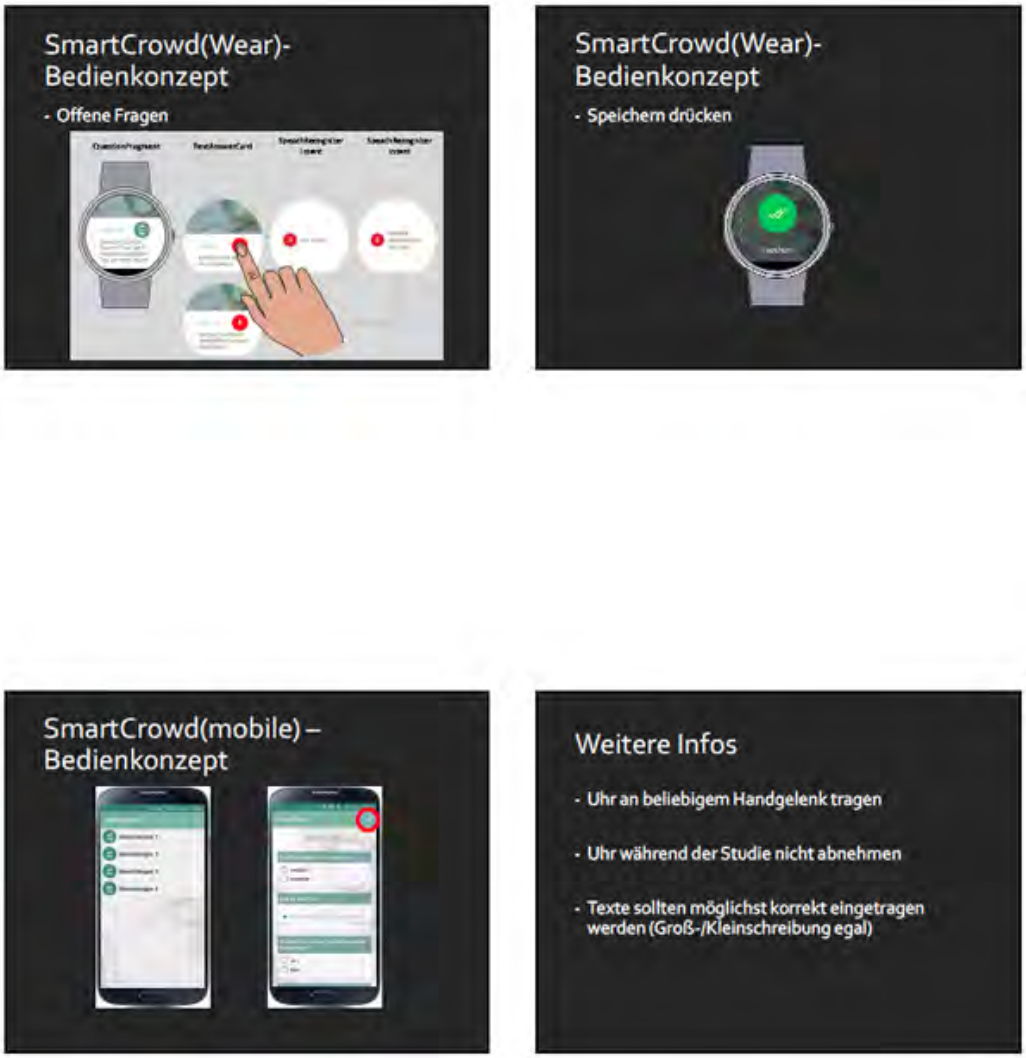

Abbildung A.3: Studieneinführung-Folien Teil 3

B

Fragebögen

Musterfragebogen

Block 1 – Teil 1:

1. Sind Sie männlich oder weiblich?
   - ○ männlich
   - ○ weiblich

2. Wie alt sind Sie?
   _____ Jahre

3. Würden Sie sich als „technikbegeistert“ bezeichnen?
   - ○ ja
   - ○ nein

4. Sind Sie im Besitz eines Smartphones?
   - ○ ja
   - ○ nein

5. Welches Betriebssystem läuft auf ihrem Smartphone?
   - ☐ Android
   - ☐ Apple iOS
   - ☐ Windows Phone
   - ☐ andere

6. Sind Sie im Besitz einer Armbanduhr?
   - ○ ja
   - ○ nein

Abbildung B.1: Musterfragebogen Seite 1

Block 1 – Teil 2

7. Welche Form muss eine Armbanduhr Ihrer Meinung nach haben?
   - o rund
   - o quadratisch

8. Sind Sie im Besitz einer Smartwatch (Armbanduhr, die über zusätzliche Computerfunktionalität verfügt)?
   - o ja
   - o nein

9. Welches Betriebssystem läuft auf Ihrer Smartwatch?
   - ☐ Android Wear
   - ☐ Apple Watch OS
   - ☐ Tizen
   - ☐ Andere

Block 2

10. Wie würden Sie Ihre Lebensqualität beurteilen? (1)
    - o Schlecht
    - o Weder gut noch schlecht
    - o Gut

11. Wie gut können Sie Ihr Leben genießen? (5)
    - o Überhaupt nicht
    - o Ein wenig
    - o Mittelmäßig
    - o Ziemlich
    - o Äußerst

12. Wie zufrieden sind Sie mit Ihrer Gesundheit? (2)
    - o Sehr unzufrieden
    - o Unzufrieden
    - o Eher unzufrieden
    - o Weder noch
    - o Eher zufrieden
    - o Zufrieden
    - o Sehr zufrieden

Block 3 (LockSlider vs. MultiButtonSlider vs. Smartphone)

13. Haben Sie genug Geld um ihre Bedürfnisse erfüllen zu können? (12)
    „Überhaupt nicht“ bis „Völlig“ (3 Stufen auf Slider)

14. Haben Sie ausreichend Möglichkeiten zu Freizeitaktivitäten?
    „Überhaupt nicht“ bis „Völlig“ (5 Stufen auf Slider)

15. Wie zufrieden sind Sie mit Ihrer Fähigkeit, alltägliche Dinge erledigen zu können?
    „Sehr unzufrieden“ bis „Sehr zufrieden“ (7 Stufen auf Slider)

Abbildung B.2: Musterfragebogen Seite 2

Block 4

16. Wie zufrieden sind Sie mit sich selbst? (18)
    - Unzufrieden
    - Weder noch
    - Zufrieden

17. Wie sicher fühlen Sie sich in Ihrem täglichen Leben? (8)
    - Überhaupt nicht
    - Ein wenig
    - Mittelmäßig
    - Ziemlich
    - Äußerst

18. Wie zufrieden sind Sie mit Ihren Wohnbedingungen? (23)
    - Sehr unzufrieden
    - Unzufrieden
    - Eher unzufrieden
    - Weder noch
    - Eher zufrieden
    - Zufrieden
    - Sehr zufrieden

Block 5

19. Haben Sie Zugang zu den Informationen, die Sie für das tägliche Leben brauchen?
    „Überhaupt nicht“ bis „Völlig“ (3 Stufen auf Slider)

20. Wie zufrieden sind Sie mit der Unterstützung durch Freunde?
    „sehr unzufrieden“ bis „sehr zufrieden“ (5 Stufen auf Slider)

21. Wie zufrieden sind Sie mit sich selbst?
    „Sehr unzufrieden“ bis „Sehr zufrieden“ (7 Stufen auf Slider)

Block 6 (Texteingabe auf der Uhr vs. Texteingabe auf Smartphone)

22. Geben Sie folgenden Satz ein: „Ich bearbeite einen Fragebogen“
    _____

23. Geben Sie folgenden Satz ein: „Ich bearbeite einen Fragebogen mit einer intelligenten Uhr“
    _____

24. Geben Sie folgenden Satz ein: „Franz jagt im komplett verwahrlosten Taxi quer durch Bayern“
    _____

Abbildung B.3: Musterfragebogen Seite 3

Vergleichsfragebogen

1. Sind Sie Rechts- oder Linkshänder?
   - ○ Rechtshänder
   - ○ Linkshänder

2. Welche Interaktionsform bevorzugen Sie auf der Smartwatch, wenn Sie eine Auswahl von Antworten zur Verfügung gestellt bekommen?

○

○

keine der beiden

○

3. Welche Interaktionsform bevorzugen Sie auf der Smartwatch für eine Skala?

○

○

keine der beiden

○

4. Haben Sie die Fortschrittsanzeige auf der linken Seite der Uhr wahrgenommen?
   - ○ ja
   - ○ nein

Fortschrittsanzeige

4.1 Falls Sie die Fortschrittsanzeige wahrgenommen haben, fanden Sie sie hilfreich?

Überhaupt nicht	Ein wenig	Mittelmäßig	Ziemlich	Äußerst
○	○	○	○	○

5. Können Sie sich vorstellen, in regelmäßigen Abständen Umfragen auf einem **Smartphone** auszufüllen?
   - ○ ja
   - ○ nein

Abbildung B.4: Vergleichsfragebogen Seite 1

6. Können Sie sich vorstellen, in regelmäßigen Abständen Umfragen auf einer **Smartwatch** auszufüllen?
   - ○ ja
   - ○ nein

7. Wieviel Fragen dürfte ein Fragebogen beinhalten, damit Sie ihn vollständig auf der Smartwatch beantworten würden?
   - ○ ≤ 3
   - ○ 4-6
   - ○ 7-9
   - ○ 10-15
   - ○ > 15
   - ○ keine Angabe

8. Können Sie sich vorstellen, Fragen zu ihrer Gesundheit, in öffentlichen Verkehrsmitteln (Zug/Bus), per Spracheingabe zu beantworten?
   - ○ ja
   - ○ nein

9. Wie anstrengend fanden Sie das Ausfüllen des Fragebogens auf der Smartwatch?

Überhaupt nicht	Ein wenig	Mittelmäßig	Ziemlich	Äußerst
○	○	○	○	○

10. Würden Sie sich gut dabei fühlen, Fragen bezüglich ihrer Gesundheit mittels einer Umfrage-App auf dem **Smartphone** zu beantworten?

Überhaupt nicht	Ein wenig	Mittelmäßig	Ziemlich	Äußerst
○	○	○	○	○

11. Würden Sie sich gut dabei fühlen, Fragen bezüglich ihrer Gesundheit mittels einer Umfrage-App auf der **Smartwatch** zu beantworten?

Überhaupt nicht	Ein wenig	Mittelmäßig	Ziemlich	Äußerst
○	○	○	○	○

12. Welche Plattform bevorzugen Sie zum Ausfüllen von Fragebögen?
   - ○ Smartphone
   - ○ Smartwatch

13. Haben Sie vor, in naher Zukunft eine Smartwatch zu kaufen?
   - ○ ja
   - ○ nein
   - ○ ich besitze bereits eine Smartwatch

Abbildung B.5: Vergleichsfragebogen Seite 2

14. Kritik und Anmerkungen bitte hier eintragen:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Abbildung B.6: Vergleichsfragebogen Seite 3

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Martin Heinzelmann Matrikelnummer: 851047

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Martin Heinzelmann